



**本科毕业设计（论文）**

**基于嵌入式Linux系统的地铁AFC设备的AGM读写器设计与实现**

|  |  |
| --- | --- |
| **学 院** | **信息工程学院** |
| **专 业** | **通信工程** |
| **年级班别** | **2019级（3）班** |
| **学 号** | **3119002470** |
| **学生姓名** | **梁丞奂** |
| **指导教师** | **郝禄国** |

2022年11月

**基于嵌入式Linux 系统的地铁AFC 设备的AGM 读写器设计与实现 梁丞奂** **信息工程学院**

摘要

智能读写器是专为轨道交通自动售检票系统应用项目设计和生产的轨道交通行业应用智能式IC卡读写模块产品。其在轨道交通自动售检票系统中有着举足轻重的地位和作用。作为票卡交互的部件，其基本功能是对票卡的读、写处理，完成设备与票卡的读写接口。如今AFC系统标准读卡器应用日趋广泛，为了适应乘客的丰富需求，其性能以及功能的进一步完善与进步是大势所趋。

本人对地铁自动检票设备中自动检票闸机进行了相关研究，并通过研究嵌入式系统的基本原理，学习嵌入式系统应用软件开发技术，掌握基于Linux的嵌入式系统软件开发的基本流程和核心技术。进而设计开发了装配在自动检票闸机上的智能读写器。本论文综合调研了目前地铁的自动检票系统、国内对智能式读写器的研究现状以及现有读卡器的提升需求，从通信技术、硬件方案、软件协议等角度对读写器进行系统的阐述。并主要从软件设计方案详细介绍了本文的研究内容和设计开发成果，最后辅于轨道交通相关的测试软件对设计的智能读写器软件进行测试。读写器的测试结果表明其能够满足不同轨道运营商系统的应用及城市通卡运营商系统的应用、满足各种新型票卡的验票交易、满足自动检票闸机验票、开关闸等业务功能的报文命令交互。

**关键词：**嵌入式，Linux系统，自动检票闸机，智能读写器，串口通信

Abstract

Intelligent read-write device is an intelligent IC card read-write module product specially designed and produced for the rail transit automatic fare collection system application project. It plays an important role in the automatic fare collection system of rail transit. As a part of ticket card interaction, its basic function is to read and write the ticket card and complete the reading and writing interface between the equipment and the ticket card. Nowadays, the standard card reader of AFC system is increasingly widely used. In order to meet the rich needs of passengers, its performance and function are further improved and improved.

I have carried out relevant research on the automatic ticket check gate in the subway automatic ticket check equipment, and learned the embedded system application software development technology through studying the basic principle of the embedded system, and mastered the basic process and core technology of the embedded system software development based on Linux. Furthermore, an intelligent read-write device installed on the automatic gate is designed and developed. This paper comprehensively investigates the current automatic ticket checking system of the subway, the research status of intelligent readers and writers in China, and the improvement requirements of existing readers, and systematically expounds the readers and writers from the perspectives of communication technology, hardware solutions, software protocols, etc. It mainly introduces the research content and design and development results of this paper in detail from the software design scheme, and finally tests the designed intelligent reader software with the help of the relevant test software of rail transit. The test results of the reader-writer show that it can meet the application of different rail operator systems and the application of the urban pass card operator systems, meet the ticket checking transactions of various new types of ticket cards, and meet the message command interaction of the business functions such as the automatic ticket check machine ticket checking, switch and so on.

**Key words**：Embedded, Linux, Automatic Gate, Intelligent Reader and Writer, Serial Communication

**目录**

[摘要 3](#_Toc129849883)

[Abstract 4](#_Toc129849884)

[1 绪论 1](#_Toc129849885)

[1.1 选题背景及目的 1](#_Toc129849886)

[1.2 国内研究现状 2](#_Toc129849887)

[1.3 论文结构及内容详细安排 3](#_Toc129849888)

[2 方案设计与基本理论 5](#_Toc129849889)

[2.1 需求分析 5](#_Toc129849890)

[2.1.1 总体功能需求 5](#_Toc129849891)

[2.1.2 模块化功能需求 6](#_Toc129849892)

[2.2 基础理论和原理分析 6](#_Toc129849893)

[2.2.1 开发环境 6](#_Toc129849894)

[2.2.2 开发工具 7](#_Toc129849895)

[2.2.3 通信技术 8](#_Toc129849896)

[2.2.4 读写技术 11](#_Toc129849897)

[2.3 方案设计 13](#_Toc129849898)

[2.3.1 整体方案设计 13](#_Toc129849899)

[2.3.2 硬件方案设计 14](#_Toc129849900)

[2.3.3 软件方案设计 16](#_Toc129849901)

[2.3.4 方案设计标准 24](#_Toc129849902)

[2.3.5 方案总结 24](#_Toc129849903)

[3 概要设计 25](#_Toc129849904)

[3.1 读写器软件概要设计 25](#_Toc129849905)

[3.1.1 公用接口子模块设计 25](#_Toc129849906)

[3.1.2运营参数模块设计 27](#_Toc129849907)

[3.1.3 自动检票闸机业务功能模块设计 28](#_Toc129849908)

[3.1.4 票卡读写模块设计 30](#_Toc129849909)

[3.2 寄存器数据设计 31](#_Toc129849910)

[3.2.1 寄存器数据存储 31](#_Toc129849911)

[3.2.2 AGM机寄存器数据设计 31](#_Toc129849912)

[4 详细设计 34](#_Toc129849913)

[4.1 软件总模块详细设计 34](#_Toc129849914)

[4.2 初始模块详细设计 40](#_Toc129849915)

[4.3公用接口模块详细设计 43](#_Toc129849916)

[4.2.1串口通信编程 43](#_Toc129849917)

[4.2.2 设备初始化 50](#_Toc129849918)

[4.2.3 版本信息获取 54](#_Toc129849919)

[4.2.4 获取SAM卡信息 54](#_Toc129849920)

[4.2.6 获取票价 54](#_Toc129849921)

[4.2.5 设备降级模式 55](#_Toc129849922)

[4.2.7 参数管理 55](#_Toc129849923)

[4.2.8 时间同步 55](#_Toc129849924)

[4.2.9 获取寄存器值 55](#_Toc129849925)

[4.2.10 获取读卡器当前信息 55](#_Toc129849926)

[4.4运营参数模块详细设计 56](#_Toc129849927)

[4.4.1参数下载 56](#_Toc129849928)

[4.4.2 参数生效通知 58](#_Toc129849929)

[4.4.3 参数加载 58](#_Toc129849930)

[4.5 AGM机业务功能模块详细设计 58](#_Toc129849931)

[4.3.1通道类型处理 58](#_Toc129849932)

[4.3.2入闸处理 60](#_Toc129849933)

[4.3.3出闸处理 61](#_Toc129849934)

[5 系统测试 63](#_Toc129849935)

[5.1 测试环境 63](#_Toc129849936)

[5.2 功能测试 63](#_Toc129849937)

[5.3 测试结果 63](#_Toc129849938)

[总结 64](#_Toc129849939)

[参考文献 64](#_Toc129849940)

[致谢 65](#_Toc129849941)

# 绪论

## 选题背景及目的

集成电路卡 (Integrated Circuit Card，IC卡)，也称智能卡(Smart Card)、智慧卡(Intelligent Card)、微电路卡(Microcircuit Card)或微[芯片](https://baike.baidu.com/item/%E8%8A%AF%E7%89%87/32249?fromModule=lemma_inlink" \t "_blank)卡等。它是将一个[微电子](https://baike.baidu.com/item/%E5%BE%AE%E7%94%B5%E5%AD%90/1410350?fromModule=lemma_inlink)[芯片](https://baike.baidu.com/item/%E8%8A%AF%E7%89%87/32249?fromModule=lemma_inlink)嵌入符合ISO7816标准的卡基中，做成卡片形式。IC卡与读写器之间的通讯方式可以是[接触](https://baike.baidu.com/item/%E6%8E%A5%E8%A7%A6/5692?fromModule=lemma_inlink" \t "_blank)式也可以是非接触式。IC卡产品应用于金融、交通、医疗、身份证明等行业，它将微电子技术和计算机技术结合在一起，提高了人们工作、生活的现代化程度。电子科技进步一日千里，在短短几年内，IC卡技术在轨道交通自动售检票（AutomaticFareCollection）系统的应用由研究摸索迅速膨胀为大规模的实际应用，非接触式IC卡以其储存量较大，保密性较强，可实现一卡多个应用的特性逐步取代了磁卡的地位，如今已成为各城市轨道交通收费系统的首选票质媒介。而智能读写器是专为轨道交通自动售检票系统应用项目设计和生产的轨道交通行业应用智能式IC卡读写模块产品。其在轨道交通自动售检票（AFC）系统中有着举足轻重的地位和作用。作为票卡交互的部件，其基本功能是对票卡的读、写处理，完成设备与票卡的读写接口。

AFC系统标准读写器在功能上需实现以下功能：实现读写器与票卡之间的通信，主要包括对票卡数据的访问和读写功能；为票卡提供所需的工作能量；实现读写器与主控设备（上位机）之间的交互通信；实现票卡的读写；具备能够存储大量数据的功能（日志、交易数据、参数文件存储）；具有防冲突机制（寻卡多卡判定）。从软件应用层面，如今的读写器从传统读写器模式逐步发展为大读写器模式。两者的本质区别是大读写器中封装了完整的票卡业务处理程序，而传统读写器的票卡处理封装在AFC设备主控程序中。这些票卡业务包括：票种、票价表等各类票卡参数的使用；车票数据读取、分析；售、补、充、进出站等各类票卡业务处理逻辑；执行各类票卡业务并产生对应的交易数据。

智能式IC卡读写模块产品具备外形小巧、功能完善、便于安装等特点，可适用于自动售检票系统终端等城市轨道交通行业终端。传统的AFC建设中技术以AFC集成厂商为主，各条线路各自招标，造成关键设备的读写器的供应商不同，硬件和软件不统一，基本不存在互换性。如今网络化建设和运营需求对关键核心设备的互换性要求提出了很高的要求：AFC智能读写器应按照线网级应用进行标准化设计和开发，票务处理软件朝着内置于读写器的方向发展，从根本上解决不同供货商设备的互通、互换，以及AFC设备适应不断发展的新票种、新业务、新应用的需求。因此，开发一种作为城市轨道交通AFC设备中的AGM机的高效智能式读卡器势在必行。

## 国内研究现状

我国城市轨道交通车站的自动售检票设备（AutomaticFareCollection），最初是来自外国，近年来我国已进行了大量的开发研制工作，提出了多种形式的产品，技术水平也在不断提高。国内轨道交通AFC系统的发展经历了从无到有的过程，随着计算机技术和软件的发展，我国城市轨道交通AFC的技术已与城市一卡通接轨，实现城市甚至城市区间的一卡通。

地铁作为现代城市的主要轨道交通工具，是大部分居民出行的优先选择。伴随着地铁的快速发展，自动售检票(AFC)设备的数量也成倍增长，运营效率极大提高的同时，日益增长的客流也对AFC设备提出了更加严峻的考验。AFC系统作为直接面向乘客的系统，能够完成自动售票、检票等基础服务。同时，AFC系统还能够进行票务管理、统计清分、财务结算等数据统计分析，为轨道交通科学合理地运营提供有力的数据支持。

基于通信、网络、计算机和自动控制等技术，自动售检票系统是一个高度自动化的系统，实现了全过程的自动化系统，包括：售检票、计费、收费、清分统计（指当日交易明细的汇总及清算净额）和管理等功能。城市轨道交通自动售检票系统的基本架构一般分为五个层次，就是通常所说的五层架构，分别为清分中心、线路中央计算机层、车站计算机层、车站终端设备层和车票五个层次。其中与乘客直接接触的主要是车票与车站终端设备，车站终端设备主要包括检票闸机、自动加值机、自动售票机与半自动售票机等，这些设备都与读卡器关系密切，尤其是AGM机即自动检票闸机。

自动检票闸机是公共场合不仅起到疏散人群的作用，同时还具备检票验票的功能。自动检票闸机是一种通道阻挡装置(通道管理设备)，用于管理人流并规范行人出入，自动检票闸机主要应用于地铁闸机系统、收费检票闸机系统、景区检票系统等、其最基本最核心的功能是实现一次只通过一人，可用于各种收费、门禁场合的入口通道处。自动检票机是闸机结合票务系统的一种具体应用。《城市轨道交通自动售检票系统技术条件》(GB／T20907--2007)规定，门式检票机在无回收票情况下通过能力不小于30AJmin。因此，理论上每台自动检票机高峰小时要通过1800位乘客。

在轨道AFC系统中，AGM机即检票闸机的功能是承担检票服务，位置往往设置在站厅付费区与非付费区之间。当乘客持单程票或储值票从非付费区进入付费区，通过闸机时，闸机上的智能读卡器就会读取车票的信息并写入入站时间、车站编号等信息，若车票满足相关规定，则给予放行。当乘客离开付费区时，闸机上的智能读卡器会再次读取车票的信息并检验其是否有效。对于有效单程票，回收后放行；对于储值票，扣除此次乘车费用后放行。往往在早晚高峰客流时，闸机的通行能力的其中一项主要影响因素就是智能读卡器的响应速率，对于市中心、客运站等节点处，高峰客流的通过需求与闸机的通过能力的矛盾更加尖锐。 

检票机是实现乘客自助进出站检票交易(在非付费区和付费区之间通行)的设备，凡持有有效车票的乘客通过检票机，其通道阻挡会自动解除(释放转杆或门扇开启)，允许乘客进出站。检票机按不同的功能可划分为进站检票机、出站检票机和双向检票机；按阻挡方式可划分为门式检票机和三杆式检票机：按通道宽带可划分普通检票机和宽道检票机(主要用于轮椅通过)。

## 论文结构及内容详细安排

本文的主要内容是基于嵌入式Linux系统的地铁AFC设备的AGM读写器的设计与实现。本文的章节层次及内容安排如下所示：

1、绪论，主要是通过介绍本课题的选题背景，并分析本课题的国内研究现状。以来设定好本论文的框架结构。

2、方案设计与基本理论，首先对本课题的需求进行分析，主要针对目前AFC设备对读写器的需求和读写器功能的需要等进行概述。其次分析本文读卡器的设计原理和基本理论。然后得出相关技术选型的方案。最后对读写器设计方案进行概述和设计标准的总结。

3、概要设计，首先划分本课题读写器的各个功能模块并画出其运行逻辑图。其次将大的功能模块划分为小的功能模块并画出其调用关系图。然后设计好数据库并画出数据库和功能模块之间的调用关系。最后对概要设计进行总结。

4、详细设计，首先对读卡器的具体业务逻辑进行具体说明并画出流程图。然后对实现该功能的代码模块进行详细说明。最后对详细设计进行总结

5、测试实现，采用地铁AFC设备中专门的测试软件对读写器各项功能进行测试并对测试结果进行分析和说明。

6、论文总结，对本课题进行全面细致的总结，指出制作本课题的需要改进的地方以及在本课题的收获。

# 方案设计与基本理论

## 需求分析

### 总体功能需求

轨道交通系统专业很多，但AFC系统的重要性不言而喻，它为乘客提供进展自动检票和出站自动售票的功能，同时也为轨道交通运营公司提供有效可靠的数据，作为科学管理的数据支持。读写器作为轨道交通AFC系统设备闸机的主要组成部分，需票具有高效、可靠、灵敏的响应能力，并须具有良好稳定的性能。即是电子收费终端部分中的智能读卡设备。智能式读写器是专为轨道交通自动售检票系统应用项目设计和生产的轨道交通行业应用智能式IC卡读写模块。该产品设计具备外形小巧、功能完善、便于安装等特点，可适用于自动售检票系统终端等城市轨道交通行业终端，也可被应用于金融IC卡接入或城市一卡通等领域。读卡器的设计应满足各种相关的技术标准，符合国际标准和国家标准。

读写器采用模块化结构设计，严格遵守轨道交通AFC标准进行设计，对外接口统一，设计的读写器满足以下需求：

**1、满足不同轨道运营商系统的应用及城市通卡运营商系统的应用。**

读写器底层预留大容量存储器，可满足现有和将来不同的应用需求，方便扩展。我方读写器是一个功能独立的、具备在安全认证模块的配合下，在读写范围内对车票完成车票分析和车票交易处理的整体功能统一的软、硬件综合体，无需外围其他设备参与读写、密钥计算功能，整个流程统一，对外接口一致。

读写器能独立完成所有与业务无关的票卡应用处理：非接触式IC卡检测、IC卡内的信息读取和写入、防冲突、选择IC卡应用、票卡密钥计算、密钥双向认证、票卡逻辑数据恢复、交易验证码产生、信用管理和交易流水号管理。

**2、满足地铁AFC设备AGM机的接入和命令报文的处理**

读写器能独立完成交易的全部处理过程，独立获取读写器运行所需要资源，与上位机程序仅使用简单接口（典型应用串口）互联，读写器与上位机仅有硬件连接，没有中间层软件。

**3、满足系统引入新功能和新票种（或新卡型）的要求**

读写器预留有新功能接口，预留手机支付功能接口，承诺满足系统引入手机支付功能和新票种（或新卡型）的要求，并可适应票卡数据结构的升级变化和跨行业的应用。

**4、满足连接智能装备并进行功能交互的要求**

智能设备是传统电气设备与计算机技术、数据处理技术、控制理论、传感器技术、网络通信技术、电力电子技术等相结合的产物。读卡器及核心模块可通过USB或串口通讯的方式与外围的智能设备进行连接,例如人脸识别、智能运维等以适合不同用户的需求。

### 模块化功能需求

**1、读卡器独立工作功能需求**

读卡器拥有强大的联网和脱机工作能力：当管理中心的监控电脑关机或通讯线路故障时，控制器能够正常地脱机工作，不会受到任何影响，在联网模式下，由管理层判断卡片是否合法，并触发例如开门动作的业务。搭载读卡器的AFC系统软件界面可弹出持卡人的照片、余额、次数等信息。

**2、接入终端设备功能需求**

获取外部模块传递的数据，从而完成外设的交互功能。能够设置设备标识，查询并识别设备，安全发现连接设备，检测通信状态，获取设备数据，对数据进行分析处理，安全断开设备，保留数据通信日志等。此需求要求读卡器有专门的串口设置。

**3、读卡器基本票务处理功能需求**

读写器能根据AGM设备的不同要求满足票卡的进出站、更新、查验等应用操作要求。读卡器用于实现对票卡的读写，在AFC系统的AGM机中大量使用。检票机读写器应能完成票卡的进出站检票交易功能。读卡器要满足将票卡处理的业务流程部署于读卡器内部，票卡业务的操作流程由读卡器独立完成，终端设备的工控机不涉及票卡业务的具体实施流程，工控机和读卡器之间只涉及操作命令、信息反馈等信息交互。

## 基础理论和原理分析

### 开发环境

[嵌入式](https://so.csdn.net/so/search?q=%E5%B5%8C%E5%85%A5%E5%BC%8F&spm=1001.2101.3001.7020)开发就是指在[嵌入式操作系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/361747)下进行开发，包括在系统化设计指导下的硬件和软件以及综合研发。除暂且分离硬件的[EDA](https://baike.baidu.com/item/EDA/182009)研发以外，侧重的就是在一定硬件条件下的系统化设计和软件研发。

**1、嵌入式开发定义：**

嵌入式开发是指利用分立元件或集成器件进行电路设计、结构设计，再进行软件编程（通常是高级语言），实验，经过多轮修改设计、制作，最终完成整个系统的开发。嵌入式技术不是单纯的软件技术，也不是单纯的硬件技术，是一门如何在一个特定的硬件环境上开发与构建特定的可编程软件系统的综合技术。

**2、嵌入式开发流程：**

当前，嵌入式开发已经逐步规范化，在遵循一般工程开发流程的基础上，嵌入式开发有其自身的一些特点。主要包括系统需求分析（要求有严格规范的技术要求）、体系结构设计、软硬件及机械系统设计、系统集成、系统测试，最终得到产品。

（1）系统需求分析。确定设计任务和设计目标，并提炼出设计规格说明书，作为正式设计指导和验收的标准。系统的需求一般分功能性需求和[非功能性需求](https://baike.baidu.com/item/%E9%9D%9E%E5%8A%9F%E8%83%BD%E6%80%A7%E9%9C%80%E6%B1%82/8403760)两方面。功能性需求是系统的基本功能，如输入输出信号、操作方式等；非功能性需求包括系统性能、成本、功耗、体积、重量等因素。

（2）体系结构设计。描述系统如何实现所述的功能和非功能性需求，包括对硬件、软件和执行装置的功能划分，以及系统的软件、硬件选型等。一个好的体系结构是设计成功与否的关键。

（3）硬件/软件协同设计。基于体系结构，对系统的软件、硬件进行详细设计。为了缩短产品开发周期，设计往往是并行的。嵌入式系统设计的工作大部分都集中在软件设计上。

（4）系统集成。把系统的软件、硬件和执行装置集成在一起，进行调试，发现并改进单元设计过程中的错误。

（5）系统测试。对设计好的系统进行测试，看其是否满足规格说明书中给定的功能要求。

### 开发工具

**1、C++**

C语言是结构化和模块化的语言，适合处理较小规模的程序。对于复杂的问题，规模较大的程序，需要高度的抽象和建模时，C语言则不合适。为了解决软件危机，20世纪80年代，计算机界提出了OOP(objectorientedprogramming：面向对象)思想，支持面向对象的程序设计语言应运而生。1982年，BjarneStroustrup博士在C语言的基础上引入并扩充了面向对象的概念，发明了一种新的程序语言。为了表达该语言与C语言的渊源关系，命名为C++。因此：C++是基于C语言而产生的，它既可以进行C语言的过程化程序设计，又可以进行以抽象数据类型为特点的基于对象的程序设计，还可以进行面向对象的程序设计。C++语言的程序开发环境，为了方便测试，将调试环境做成了解释型。即开发过程中，以解释型的逐条语句执行方式来进行调试，以编译型的脱离开发环境而启动运行的方式来生成程序最终的执行代码。

开发C++应用程序，需要经过编写源程序、编译、连接程序生成可执行程序、运行程序四个步骤。生成程序是指将源码（C++语句）转换成一个可以运行的应用程序的过程。如果程序编写正确，那么通常只需按一个功能键，即可完成该过程。

第一步对程序进行编译，这需要用到编译器（compiler）。编译器将C++语句转换成机器码（也称为目标码）；如果该步骤成功执行，下一步就是对程序进行链接，这需要用到链接器（linker）。链接器将编译获得机器码与C++库中的代码进行合并。C++库包含了执行某些常见任务的函数（“函数”是子程序的另一种称呼）。例如，一个C++库中包含标准的平方根函数sqrt，所以不必亲自计算平方根。C++库中还包含一些子程序，它们把数据发送到显示器，并知道如何读写硬盘上的数据文件。

2、ARM平台

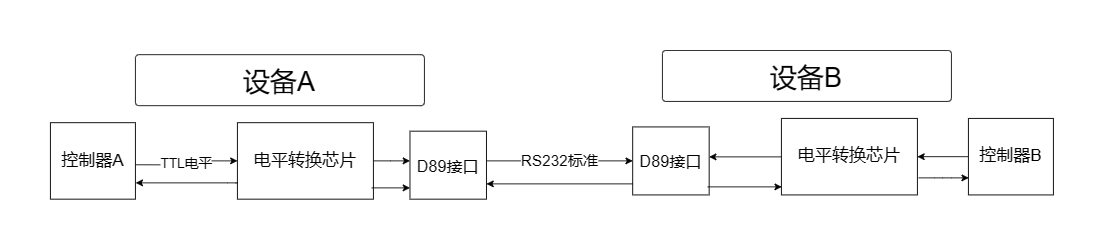
### 通信技术

**1、串口通信简介**：

串口通讯(SerialCommunication)是一种设备间非常常用的串行通讯方式，因为它简单便捷，因此大部分电子设备都支持该通讯方式，电子工程师在调试设备时也经常使用该通讯方式输出调试信息。

在计算机科学里，大部分复杂的问题都可以通过分层来简化。如芯片被分为内核层和片上外设；STM32标准库则是在[寄存器](https://so.csdn.net/so/search?q=%E5%AF%84%E5%AD%98%E5%99%A8&spm=1001.2101.3001.7020" \t "_blank)与用户代码之间的软件层。对于通讯协议，我们也以分层的方式来理解，最基本的是把它分为物理层和协议层。物理层规定通讯系统中具有机械、电子功能部分的特性，确保原始数据在物理媒体的传输。协议层主要规定通讯逻辑，统一收发双方的数据打包、解包标准。简单来说物理层规定我们用嘴巴还是用肢体来交流，协议层则规定我们用中文还是英文来交流。

RS-232标准接口（又称EIARS-232）是常用的串行通信接口标准之一



**2、电平标准**

在上面的通讯方式中，两个通讯设备的“DB9接口”之间通过串口信号线建立起连接，串口信号线中使用“RS-232标准”传输数据信号。由于RS-232电平标准的信号不能直接被控制器直接识别，所以这些信号会经过一个“电平转换芯片”转换成控制器能识别的“TTL标准”的电平信号，才能实现通讯。

根据通讯使用的电平标准不同，串口通讯可分为TTL标准及RS-232标准，TTL电平标准与RS232电平标准。

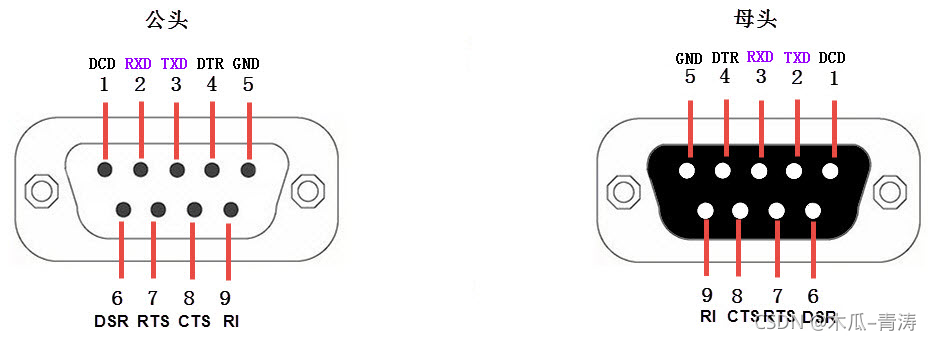
|  |  |
| --- | --- |
| 通信标准 | 电平标准 |
| 5V TTL | 逻辑1：2.4~5V  逻辑0：0~0.5V |
| RS-232 | 逻辑1：-15~3V  逻辑0：3~15V |

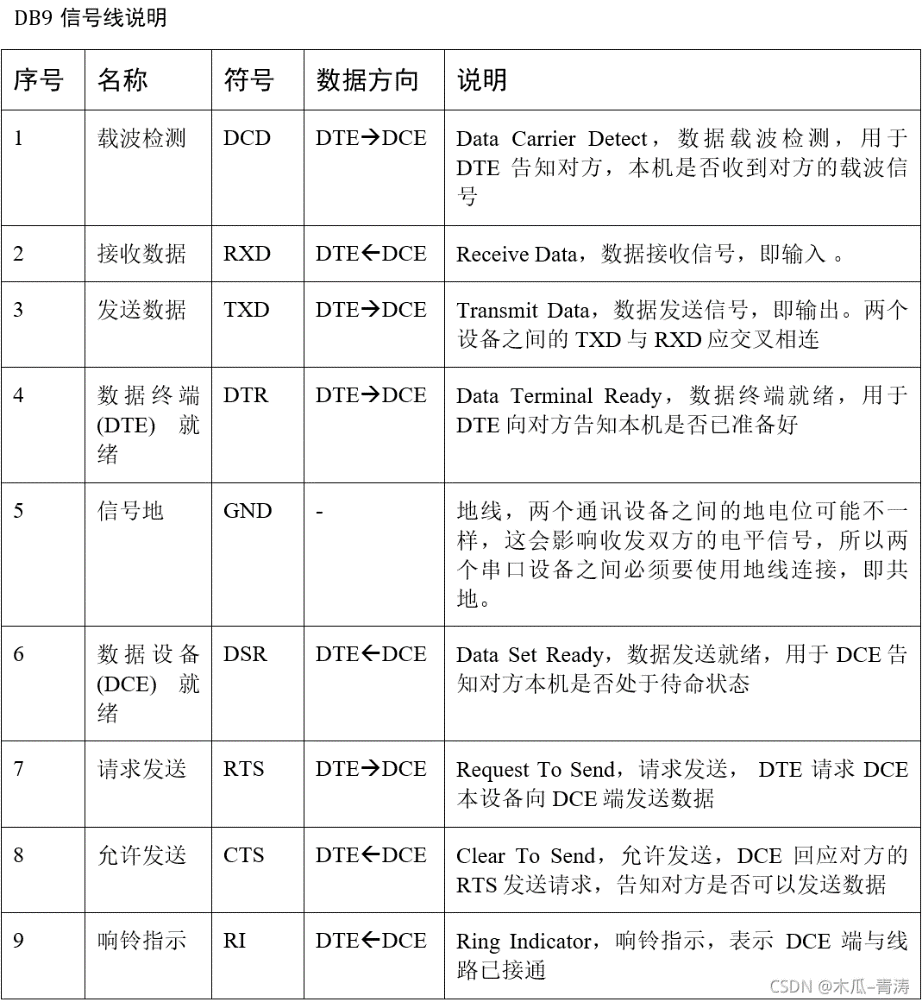
电子电路中常使用TTL的电平标准，理想状态下，使用5V表示二进制逻辑1，使用0V表示逻辑0；而为了增加串口通讯的远距离传输及抗干扰能力，它使用-15V表示逻辑1，+15V表示逻辑0。使用RS232与TTL电平校准表示同一个信号时的对比见图RS-232与TTL电平标准下表示同一个信号。

**3、信号线**

RS-232串口标准常用于计算机、路由与调制调解器(MODEN，俗称“猫”)之间的通讯，在这种通讯系统中，设备被分为数据终端设备DTE(计算机、路由)和数据通讯设备DCE(调制调解器)。我们以这种通讯模型讲解它们的信号线连接方式及各个信号线的作用。在旧式的台式计算机中一般会有RS-232标准的COM口(也称DB9接口)。



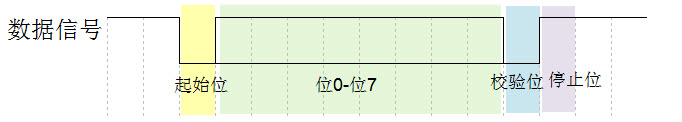




**4、协议层**

串口通讯的数据包由发送设备通过自身的TXD接口传输到接收设备的RXD接口。在串口通讯的协议层中，规定了数据包的内容，它由启始位、主体数据、校验位以及停止位组成，通讯双方的数据包格式要约定一致才能正常收发数据，其组成见图串口数据包的基本组成。

串口数据包的基本组成



（1）波特率：数据信号对载波的调制速率，串口异步通讯中由于没有时钟信号(如前面讲解的DB9接口中是没有时钟信号的)，所以两个通讯设备之间需要约定好波特率，即每个码元的长度，以便对信号进行解码，图串口数据包的基本组成中用虚线分开的每一格就是代表一个码元。常见的波特率为4800、9600、115200等。

（2）通讯的起始和停止信号：串口通讯的一个数据包从起始信号开始，直到停止信号结束。数据包的起始信号由一个逻辑0的数据位表示，而数据包的停止信号可由0.5、1、1.5或2个逻辑1的数据位表示，只要双方约定一致即可。

（3）有效数据：在数据包的起始位之后紧接着的就是要传输的主体数据内容，也称为有效数据，有效数据的长度常被约定为5、6、7或8位长。

（4）数据校验：在有效数据之后，有一个可选的数据校验位。由于数据通信相对更容易受到外部干扰导致传输数据出现偏差，可以在传输过程加上校验位来解决这个问题。校验方法有奇校验(odd)、偶校验(even)、0校验(space)、1校验(mark)以及无校验(noparity)。  
奇校验要求有效数据和校验位中“1”的个数为奇数，比如一个8位长的有效数据为：01101001，此时总共有4个“1”，为达到奇校验效果，校验位为“1”，最后传输的数据将是8位的有效数据加上1位的校验位总共9位。偶校验与奇校验要求刚好相反，要求帧数据和校验位中“1”的个数为偶数，比如数据帧：11001010，此时数据帧“1”的个数为4个，所以偶校验位为“0”。0校验是不管有效数据中的内容是什么，校验位总为“0”，1校验是校验位总为“1”。

### 读写技术

**1、NFC简介**

近场通信（NearFieldCommunication，简称NFC），是一种新兴的技术，使用了NFC技术的设备（例如移动电话）可以在彼此靠近的情况下进行数据交换，是由非接触式[射频](https://so.csdn.net/so/search?q=%E5%B0%84%E9%A2%91&spm=1001.2101.3001.7020" \t "_blank)识别（[RFID](https://baike.baidu.com/item/RFID/497249)）及互连互通技术整合演变而来的，通过在单一芯片上集成感应式读卡器、感应式卡片和点对点通信的功能，利用移动终端实现移动支付、电子票务、门禁、移动身份识别、防伪等应用。NFC可以理解为是RFID的一个子集。目前手机比较火的“碰一碰”功能就依托于该技术。NFC只是提供了一种简单、触控式的解决方案，可以让客户直观地和智能设备交换信息、访问所需内容、提供相应的服务。NFC技术支持设备之间进行非接触式点对点的数据传输，可以在10cm距离内交换数据，其传输速度主要有106Kbit/秒、212Kbit/秒或者424Kbit/秒三种。

**2、NFC工作模式**

**（1）**点对点模式(P2Pmode)和红外线差不多，用于数据交换，该模式传输距离较短，传输速度较快，功耗低（和蓝牙也类似）。将两个具备NFC功能的智能设备配对连接，能实现数据点对点传输，典型的例子比如智能手机自带的NFC可以让两台手机之间快速传输文件。

**（2）**有卡模式(Cardemulation)相当于一张采用RFID技术的IC卡。可以替代大量的IC卡（包括信用卡）、商场刷卡、交通卡、门禁，火车票等等。有卡模式有一个很大的优势就是卡片通过非接触读卡器的RF域进行供电，即便是寄主设备没电也可以工作。

**（3）**读卡器模式(Reader/writermode)其实就是通过支持NFC的智能手机或其它电子设备从带有NFC芯片的标签、贴纸、名片等媒介中读写信息。通常NFC标签是不需要额外外部供电的。当支持NFC的外设向NFC读写数据时，它会发送某种磁场，而这个磁场会自动向NFC标签供电。

**3、卡片读写标准**

[ISO/IEC14443](https://baike.baidu.com/item/iso14443/1967008)定义了TypeA、TypeB两种卡片读写标准：TYPEA卡在读写机上向卡传送信号时，是通过13.56MHz的射频载波传送信号。其采用方案为同步、改进的Miller编码方式，通过100%ASK传送；当卡向读写机具传送信号时，通过调制载波传送信号。使用847kHz的副载波传送Manchester编码。而TYPEB卡在读写机具向卡传送信号时，也是通过13.56MHz的射频载波信号，但采用的是异步、NRZ编码方式，通过用10%ASK传送的方案；在卡向读写机具传送信号时，则是采用的BPSK编码进行调制。对比两种卡型，可以看出，TypeB型与TypeA型卡相比有以下优势：

（1）芯片具有更高的安全性。接收信号时，不会因为能量损失而使芯片内部逻辑及软件工作停止。

（2）支持更高的通讯速率。TypeA最大的数据通讯速率为150Kbit/s-200Kbit/s，应用10%ASK技术的TypeB至少可支持400Kbit/s的速率。

（3）外围电路设计简单。读写机具到卡以及卡到读写机具的编码方式均采用NRZ方案，电路设计对称，设计时可使用简单的UARTS。

（4）抗干扰能力强。负载波采用BPSK调制技术，较TypeA方案降低了6dB的信号声。

在ISO/IEC14443-3规定了TYPEA，TYPEB的防冲突机制。二者防冲突机制的原理完全不同。前者是基于BIT冲突检测协议，后者则是通过字节、帧及命令完成防冲突。防冲突机制使非接触式IC卡能进行并行操作，及在多张卡同时进入有效操作区后，可对其进行有条不紊的操作，这样就使选定卡片的数据不受其它卡数据干扰，携带有多种卡的用户可不必寻找正确的一张卡，只用算法编程，读写机具即可自动做到选取正确的一张卡进行后续操作。这样方便了操作，提高了应用的并行性，也提高了系统的速度。

## 方案设计

### 整体方案设计

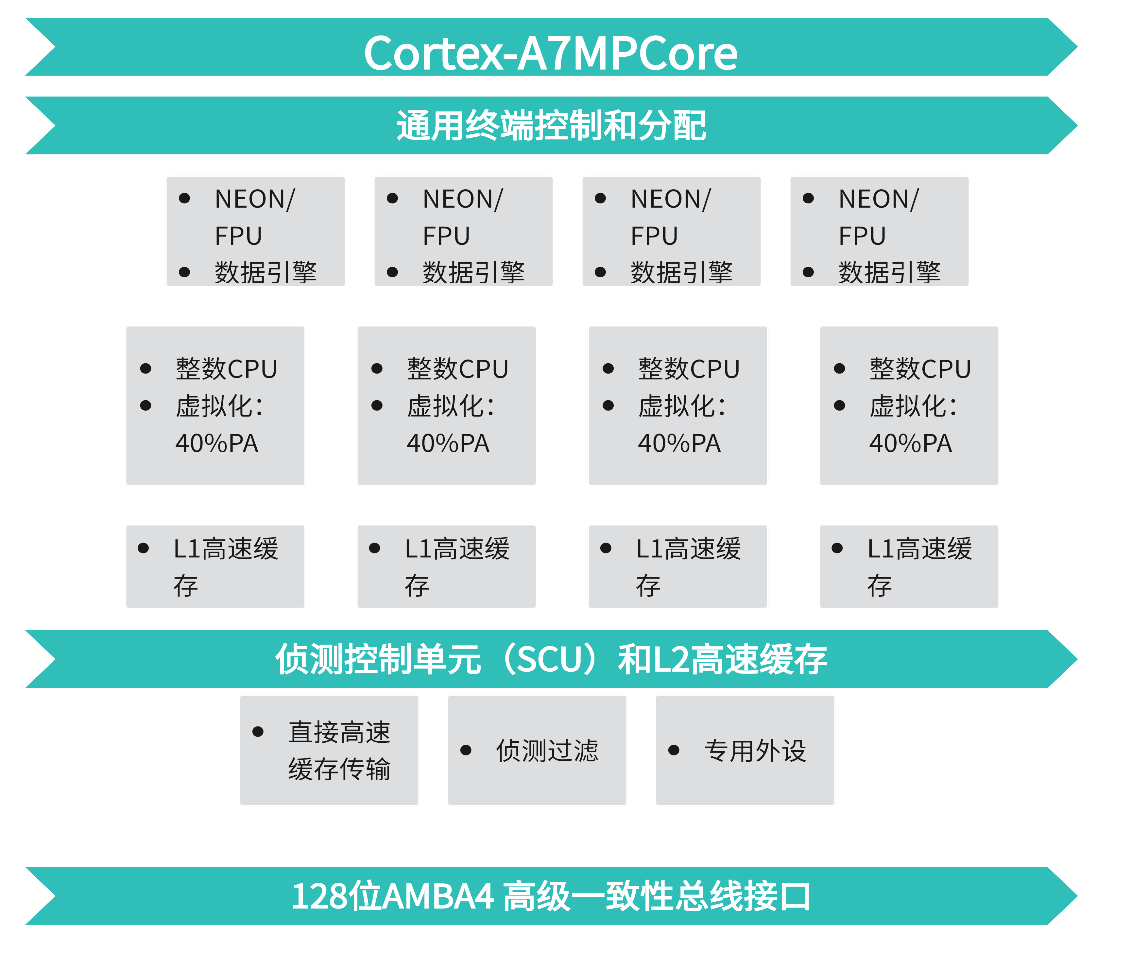
整体设计思想：根据需求分析，应用于轨道交通AFC系统的专用读卡器对体积、功耗和系统的可靠性均有较为严格的要求，而嵌入式系统正好可以满足AFC系统专用读卡器的这些要求。对嵌入式系统的定义是：以应用为中心、以计算机技术为基础、软硬件可裁剪、适应应用系统对功能、可靠性、成本、体积、功耗严格要求的专用计算机系统。

从嵌入式系统的定义可以明显地得出其主要优点符合读卡器的设计要求，所以采用嵌入式平台对AFC系统专用读卡器进行设计。嵌入式系统具有嵌入性、系统性和智能性，具体来说嵌入式系统是嵌入到特定对象体系中的实体，是一个计算机系统同时具有嵌入式软硬件并能智能地实现预定的功能。专用性、实时性是嵌入式系统的主要特点的组成部分，专用性主要体现在嵌入式的专用处理器上，根据能、实现的功能不同有不同的专用处理器，如视频处理系统、语言处理系统等；实时性是因为嵌入式系统常被用作控制或者辅助操作设备的装置，因此要求其具有很好的实时性来满足对象系统的要求；除此之外，嵌入式系统还具有系统设备电子化、编程语言低级化、系统性能可测化以及开发系统专用化等特点。在嵌入式系统中，系统的控制核心是处理器，应具备良好的性能。

### 硬件方案设计

**1、读卡器芯片方案**

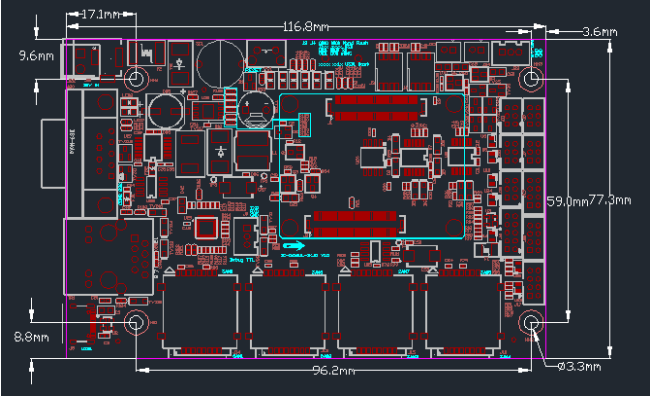
主控电路板选用ARMCortex-A7为核心CPU的MX6UL-C微控制器。ARMCORTEX-A7是一款经典的工业级CPU，其优越稳定的性能、高性价比，调试简便，外设接口丰富等特征使得该CPU被广泛地应用于各行各业。

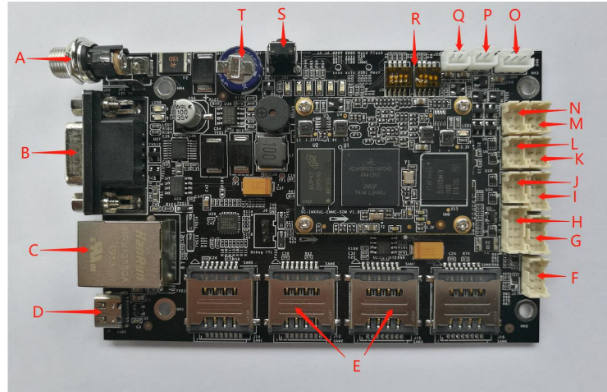


由于ARMCortex-A7有着良好的性价比，其功能及性能可满足绝大多数的应用场合，生产工艺要求相对简单。该CPU有着巨大成熟的应用市场，普及程度高，有旺盛的生命力。根据的需求分析，本方案选用MX6UL-C作为读写器的主控制器。

**2、读卡器PCB方案**

|  |  |
| --- | --- |
| **项目** | **配置和性能参数** |
| 中央处理器 | ARMCortex-A7 |
| 主频 | 528MHz |
| ROM | 4GB |
| RAM | 512MB |
| 天线模块 | 3个 |
| SAM卡槽 | 8个 |
| 串口 | TTL电平：6路 RS232电平：5路 |
| USB | Host：1路 OTG：1路 |
| 网口 | 1个· |
| SD卡槽 | 1个· |
| 铁电存储 | 1个· |
| 加密芯片 | 1个· |
| 蜂鸣器 | 1个 |
| 供电电源 | DC12V |
| 功耗 | ≤3W |
| 存储温度 | -40℃~125℃ |
| 运行环境温度 | -20℃~70℃ |
| 相对湿度 | 10%~90% 无凝露 |





|  |  |  |  |
| --- | --- | --- | --- |
| A | 电源接口 | F、G、I、H | 天线接口 |
| B | DB9 通讯口 | J、K、L | 串口 TTL |
| C | RJ45 网络接口 | N | 串口RS232(ttymxc4/5) |
| D | USB OTG 接口 | R | 拨码开关 |
| E | 8 个 SAM 卡槽 | S | 复位按键 |

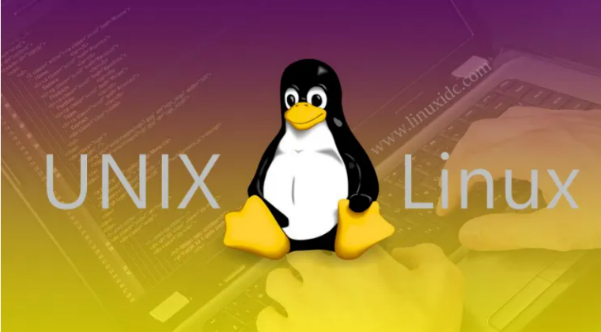
**3、读卡器天线方案**

读卡器选用外置天线，通过精心设置天线的形状和大小，使之在符合安装要求有着良好的性能，选择天线合适的品质因数Q，精确实现天线的阻抗匹包括线网系统中正在使用的票卡和在建线路采购的票卡品质因数。天线采用特征阻抗为50Ω的同轴电缆与读卡器相连系统专用读卡器的设计与实现不通过处理很难得到理想的电磁频率，所以读为了使EMC标EMC的方法是设是高频巴伦变压器，要连实现平衡到不平衡转换。使之在符合安装要求精确实现天线的阻抗匹包括线网系统中正在使用的票卡和在建线路采购的票卡）的同轴电缆与读卡器相连。

### 软件方案设计

**1、读卡器整体软件架构**

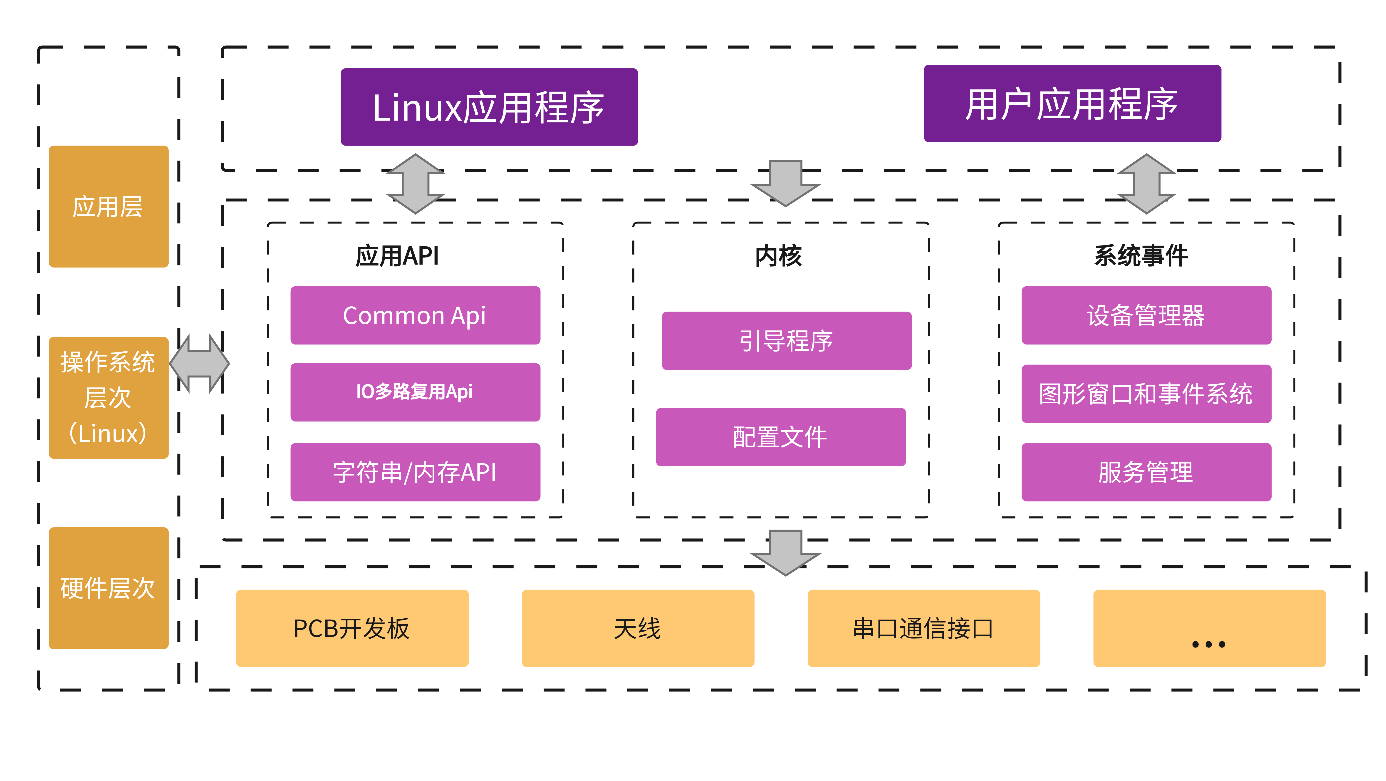
读卡器的功能要求比较复杂，需要多任务处理，且对系统的安全性和稳定性要求较高。由于嵌入式操作系统拥有完整的编程规范和统一的接口函数，可使用高级程序语言，从而使应用软件具有良好的代码可重用性和可移植性；同时，嵌入式操作系统具有多任务处理功能，能快速响应外部中断任务，能够显著地提高读卡器系统的整体性能。在选择读卡器操作系统时，主要考虑如下因素：(1)选用主流的、成熟的操作系统，该操作系统需拥有强盛的生命力，已经被广泛接受，将在可预见的未来，仍为主流操作系统。(2)应用软件可实现与城市轨道交通不同线路之间的兼容。(3)尽可能选用免费或低成本的系统，便于未来的大规模生产发布。



嵌入式Linux是将日益流行的Linux操作系统进行裁剪修改，使之能在嵌入式计算机系统上运行的一种操作系统。因此嵌入式Linux具备了Linux系统开源的优点，也具备了嵌入式这一特征。更重要的原因之一是其为免费系统，而且具有优异的性能，因其为开源代码的特点，软件移植起来相对容易，且应用软件开发周期短，具备良好的安全性、实时性、稳定性。由于市场发展前景广阔，Montavista、Lineo、Emi等专业公司开发了许多基于嵌入式Linux的专用产品，有行业协会如EmbeddedLinuxConsortum等，而IBM、Motorola、Intel等知名软件公司也纷纷支持，还有Internet上的大量嵌入式Linux爱好者的支持。嵌入式Linux几乎支持所有的嵌入式CPU，同时几乎可以被移植到所有的嵌入式OEM板。可见嵌入式Linux在可预计的未来，将保持着旺盛的生命力。

综上所述，嵌入式Linux具有诸多优点：首先，Linux因其开源的特性，得到了全球的众多专业厂家及编程爱好者的支持；其次，Linux的内核只有约134KB大小，具有执行效率高，更新速度的特点，而且也是可以定制的。第三，Linux系统因其免费的特点，具备价格优势。Linux适用于嵌入式操作系统还体现在，它是一个跨平台的系统，截止目前，有二三十种CPU和硬件平台的支持。具有良好的稳定性，裁剪性，开发和使用的便利性。其移植速度远远超过以前的Java系统，因此越来越多的CPU芯片厂商纷纷加入这一行列。同时，Linux在网络方面的完整性得益于其内核的结构，甚至于提供了以及无线网络，令牌环网(Tokerring)、光纤甚至卫星的支持。嵌入式Linux提供了包括十兆、百兆、千兆的以太网络，完全支持目前应用最为广泛的网络TCP/IP协议。保证了目前使用Linux环境开发的产品，在将来的适用性。

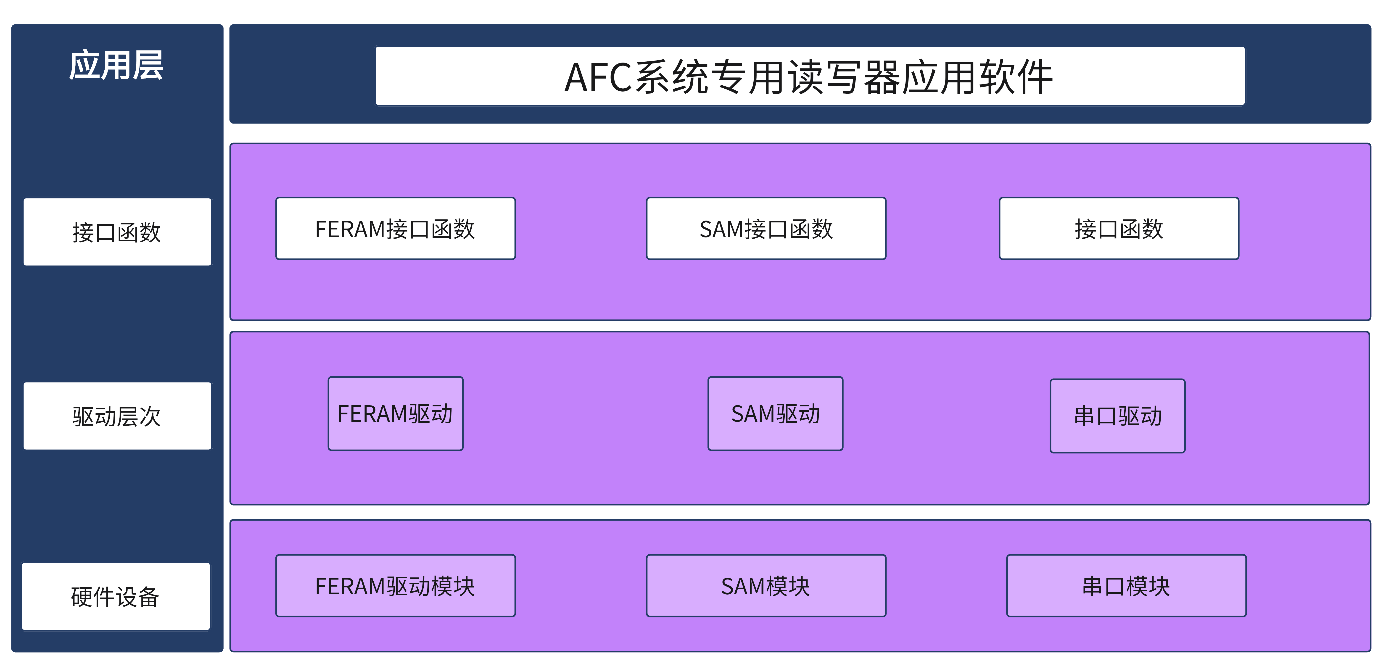
读写器选用Linux系统，并在此系统上开发驱动和应用程序。Linux操作系统有两种空间状态，各个功能硬件设备的驱动位于Linux的内核空间，而其他应用软件在其用户空间。其具体软件架构层次如图所示：



嵌入式 Linux 系统体系结构基于层次化思想进行设计，其分层架构模型如图所示，从底层向上依次为硬件层、OEM、操作系统层和应用层。硬件层是指系 统的硬件组成，包括微处理器、各种外围设备和硬件功能模块；OEM 层位于硬件层和操作系统层之间，为二者的交互提供接口，操作系统可以通过OEM适配层提供的应用接口程序访问具体的硬件，从而避免了直接与硬件进行匹配。操作系统层 位应用程序提供载体，可以根据具体的用户需求定制 Linux 操作系统的组件，保留需要的组件，删除不必要的组件，合理的剪裁可以优化系统的性能。应用层包括 Linux 应用程序和用户应用程序，是用户为特定的系统开发的个性化应用。

**2、 操作系统及底层驱动方案**

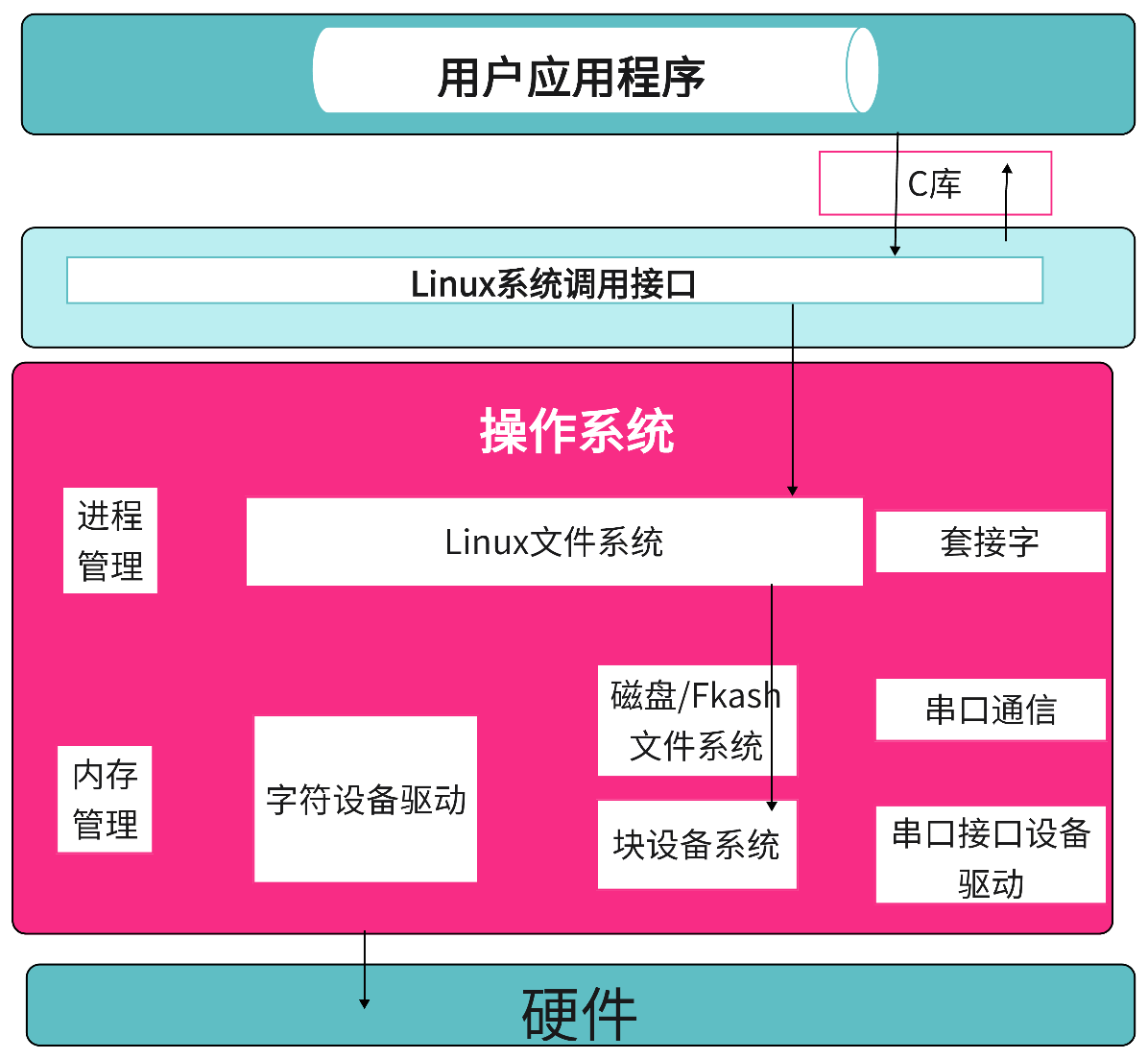
根据Linux的层次体系架构，结合AFC系统专用读写器系统的具体情况，可以得出AFC系统专用读写器系统的架构，大多数功能硬件需要开发驱动，除去自身封装有驱动程序的硬件，主要的硬件驱动程序与操作系统以及应用软件的结构关系如图4-2所示，其中不包括LED灯等次要硬件及其驱动程序。



应用软件需要采用C/C++语言开发，代码只采用标准的ANSI函数库和POSIX函数库，这样程序具有很强的可移植性。软件能根据不同的操作系统、开发环境的变化而重新独立编译或移植。

设备驱动程序负责将操作系统的请求传输转化为特定物理设备控制器能够理解的命令，通过读写硬件寄存器实现对硬件设备的控制。设备驱动程序是操作系统和输入输出设备之间的沟通桥梁，它使计算机能够认识和识别这些硬件设备，从而保证硬件设备的正常运行。设备驱动程序是一种内核模块，负责管理硬件设备底层I/O操作，内核通过调用标准接口函数与设备进行交互。

在Linux中，为了简化对设备的管理，所有外围的硬件设备被归结为三类：字符设备、块设备和网络设备。字符设备是指鼠标、键盘、串行口等，存取时没有缓存的设备；块设备是指硬盘设备、FLASH等，读写都有缓存支持，并能随机存取；网络设备在Linux里做专门处理，系统里支持对发送数据和接收数据的缓存，但不能通过简单的读写操作来访问它们，如以太网卡等。除了网络设备外，字符设备和块设备都被映射到Linux文件系统的文件和目录中，访问字符设备和块设备需通过文件系统的调用接口函数open、write、read、close实现。而块设备较字符设备复杂，在块设备上边首先会建立一个磁盘/FLASH文件系统，不同的设备对应的文件系统也不同，例如硬件中NORFLASH采用JFFS2文件系统，NANDFLASH采用YAFFS2文件系统。如图Linux设备驱动与整个软硬件系统的关系。



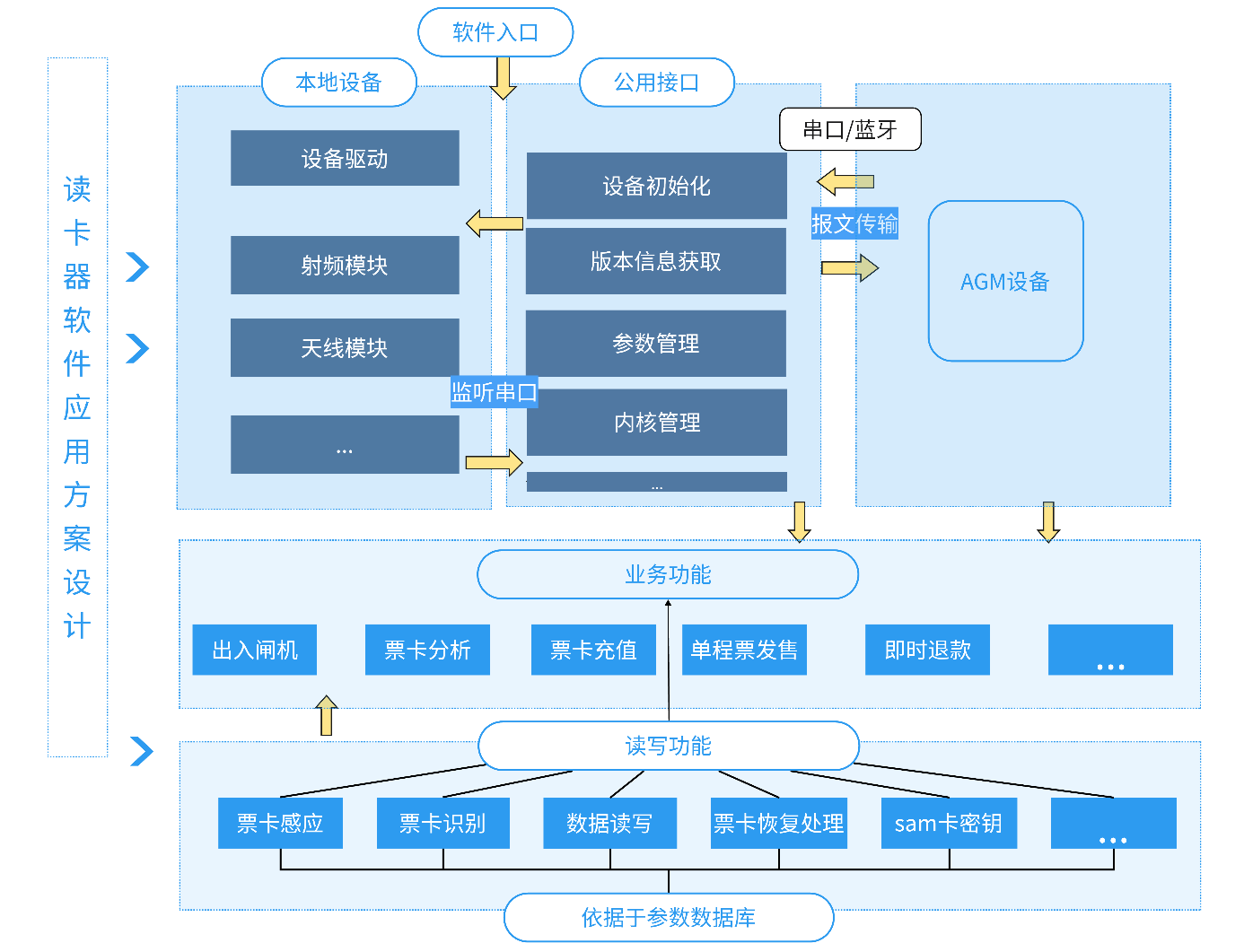
Linux将所有外部设备看成是一类特殊文件，称之为设备文件，设备驱动程序相当于是Linux内核与外部设备之间的接口。设备驱动程序简化了应用程序对硬件的操作，实现应用程序操作硬件时与正常文件操作没有特殊不同。设备驱动程序的主要功能包括：设备初始化和释放；设备的参数设置以及操作设备的接口；实现应用程序与设备文件之间数据的传送和返回请求；实现对设备的监测及纠错处理。

Linux操作系统对所有的设备的操作是通过文件的操作界面进行的，而设备文件的属性由三部分信息组成：文件的类型、主设备号和次设备号。文件类型和主设备号两者共同保证了设备文件的驱动程序唯一性，也就是说主设备号相同的设备使用相同的驱动程序，次设备号只是用于说明目标设备在同类设备的位次。Linux操作系统按所存空间不同分为内核空间和用户空间。内核空间是指在被保护的内存空间上的核心软件，拥有较高的级别，拥有所有访问硬件设备的权限；用户空间则为在用户空间执行其他部分的应用程序。Linux系统实现从用户空间到内核空间的转移主要是通过系统调用和硬件中断完成的，大量的设备驱动程序及相应的应用软件就是采用内核级与用户级的程序一起来完成驱动与访问的。驱动程序是运行于内核空间的，用户空间的应用程序通过文件系统中的一个设备文件与其交互。

**3、读卡器软件应用方案**

通过对AFC系统专用读卡器在实际中应用的研究，发现读卡器软件技术在设计与开发的过程中主要有四个方面。（1）需要将动态库技术应用于系统之中，这种技术能够实现程序空间的有效压缩，从而促进程序执行效率的提高，使其运行具有较高的灵活性。动态库技术的应用能够使其使用更加简便；（2）需要将软件进行参数化处理，参数化主要体现在两个方面，分别为票务规则参数化、软件自身的参数化，通过参数的设置，来进行不同选项的配置，来满足系统在功能方面的需求；（3）需要保证系统数据的安全性，这一点在软件开发全过程中都是需要进行重点考虑的环节；（4）系统具备兼容性，这种性能保证了读卡器软件自身运行具有的时效性，使城市轨道交通是AFC系统在长时间的运行状态下的需求能够得到满足。

轨道交通AFC系统专用读卡器方案要得以实现，必须要从驱动程序与票卡交易两方面的实现入手。驱动程序的主要作用就是转化，当操作系统进行信息请求与传输的过程中，通过转化的作用将请求转化为物理设备控制器，这种特定的控制器能够对请求命令更加方便的进行理解，并对硬件寄存器进行读卡以达到对硬件设备进行控制的目的。票卡交易主要作用是实现各个载体的通信，每一个载体的功能都不同，比如车站进出口的闸机、用于售票的自动与半自动售票机以及自动加值机等等，票卡就是在车站中的这些机器载体中来实现其通讯目的的。



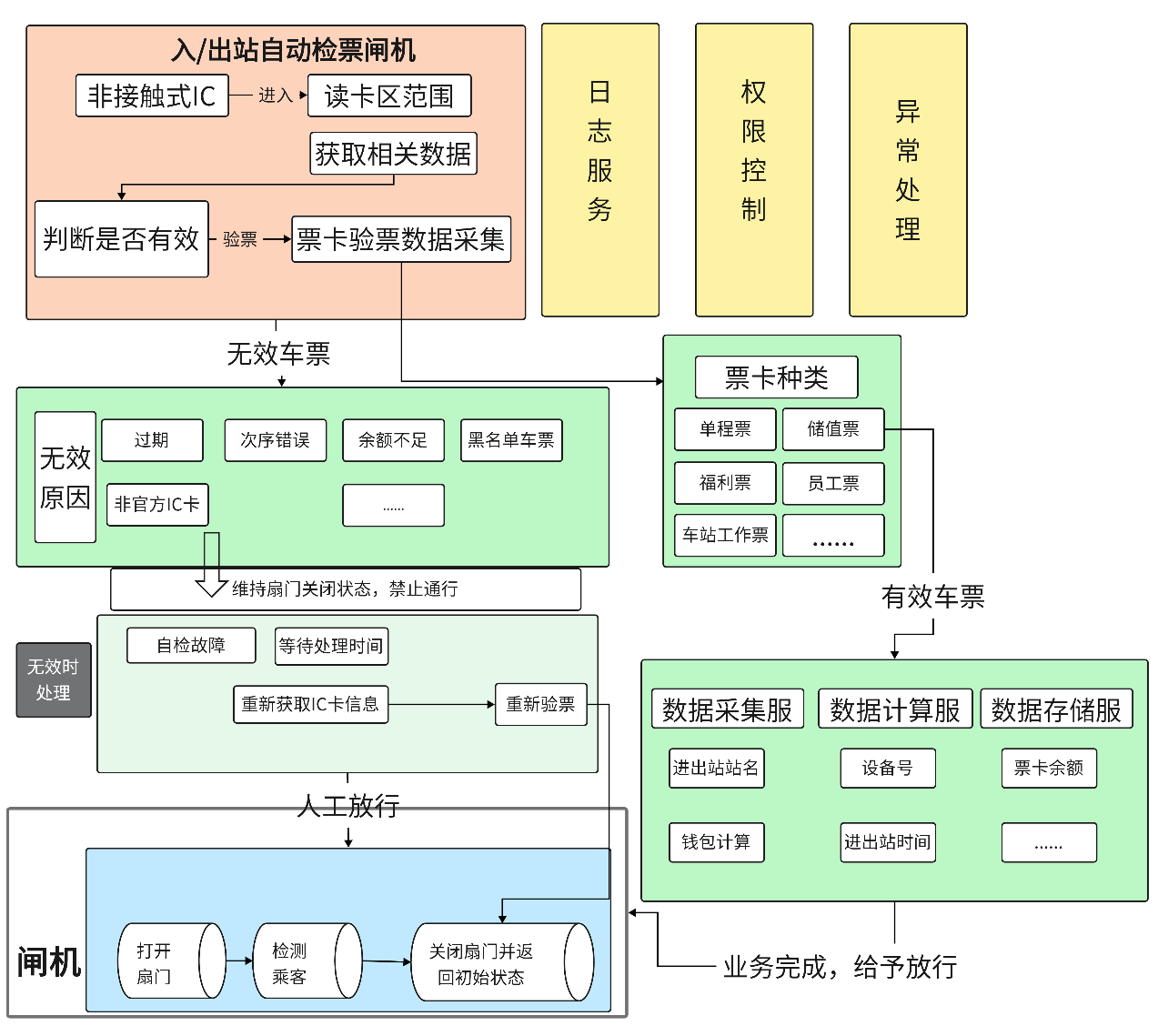
AFC系统专用读写器在城市轨道交通AFC系统中主要应用在闸机(自动检票机)、自动售票机、半自动售票机以及自动加值机上，通过这些载体与票卡进行通讯。AFC系统专用读写器的作用就是监管、控制或者更改票卡里的特定数据从而帮助实现AFC系统对城市轨道交通的票务管理。根据以上设备的功能可知票卡交易的过程有充值、进站和出站等。读写器的功能是与票卡之间实现有效可靠的交易，所有读写器的应用层软件的设计就是围绕读写器这一功能进行的。

在轨道AFC系统中，AGM机即检票闸机的功能是承担检票服务，位置往往设置在站厅付费区与非付费区之间。当乘客持单程票或储值票从非付费区进入付费区，通过闸机时，闸机上的智能读卡器就会读取车票的信息并写入入站时间、车站编号等信息，若车票满足相关规定，则给予放行。当乘客离开付费区时，闸机上的智能读卡器会再次读取车票的信息并检验其是否有效。对于有效单程票，回收后放行；对于储值票，扣除此次乘车费用后放行。

基于需求本软件设计主要分为五大模块：首先是调动读卡器PCB板射频、天线等模块的驱动程序，能够对读卡器开发板的数据进行读写。其次是公用接口模块，用于与AGM机进行报文的通信交互。然后是AGM机业务功能模块，能够在接收AGM报文命令之后，实现开关出入闸机、识别票卡、票卡分析等业务功能。再然后是读写模块，能够读取票卡的相关数据回送至AGM机，也能够对票卡的数据进行改写以达成相关的业务功能。最后是寄存器模块，具有存储本地数据、进行参数下载以及储存各种数据格式等功能。

**4、读卡器软件方案流程**

基于AGM闸的读卡器软件设计和实现，本智能读卡器软件方案业务流程如下：



（1）进检票机工作流程：使用一张非接触式IC卡进入读卡区范围，读卡器将对车票进行有效性检查，若为有效票，则自动将进站站名、进站时间和设备号等信息写入车票中，然后打开扇门，检测到乘客通过后关闭扇门并返回到开始状态，若为无效票（车票无效条件：过期、次序错误、余额不足、黑名单车票、非官方发行的IC卡等），则提示车票无效或报警，并维持扇门关闭状态禁止通行。

（2）出站检票机工作流程：使用一张非接触式IC卡进入读卡区范围，对车票进行有效性和车费检查，若为有效票，按以下票卡种类进行处理：单程票、福利票、出站票则自动写入注销信息并回收；定值票、储值票、计次票等扣除相应乘车费用和乘次：员工票、车站工作票等免费车票写入相应记录。处理完成之后打开扇门，检测到乘客通过后关闭扇门并返回到开始状态；若为无效票或费用不够，则提示无效或欠费，并维持扇门关闭状态禁止通行。

（3）双向检票机工作流程：具备进站检票机和出站检票机两种功能，可设置为进站检票机状态、出站检票机状态、或是进/出站检票机状态。当检票机处于进站状态时，设备自动执行进站检票机的工作流程；当检票机处于出站状态时，设备自动执行出站检票机的工作流程。

（4）降级模式票卡处理流程：

AFC系统处于不同运行模式时， AGM闸机对车票的处理情况如下：

紧急放行模式：全部闸机扇门处于全开状态，顶棚向导标志处于禁入放行状态，乘客出站不检票。

列车故障模式：全部闸机扇门处于关闭状态，进站闸机正常检票入站；出站闸机检票出站不扣费，除福利票、出站票外回收类车票不回收。

进站免检模式：进站闸机扇门处于全开状态，乘客进站不检票；出站闸机关闭状态，正常检票出站。

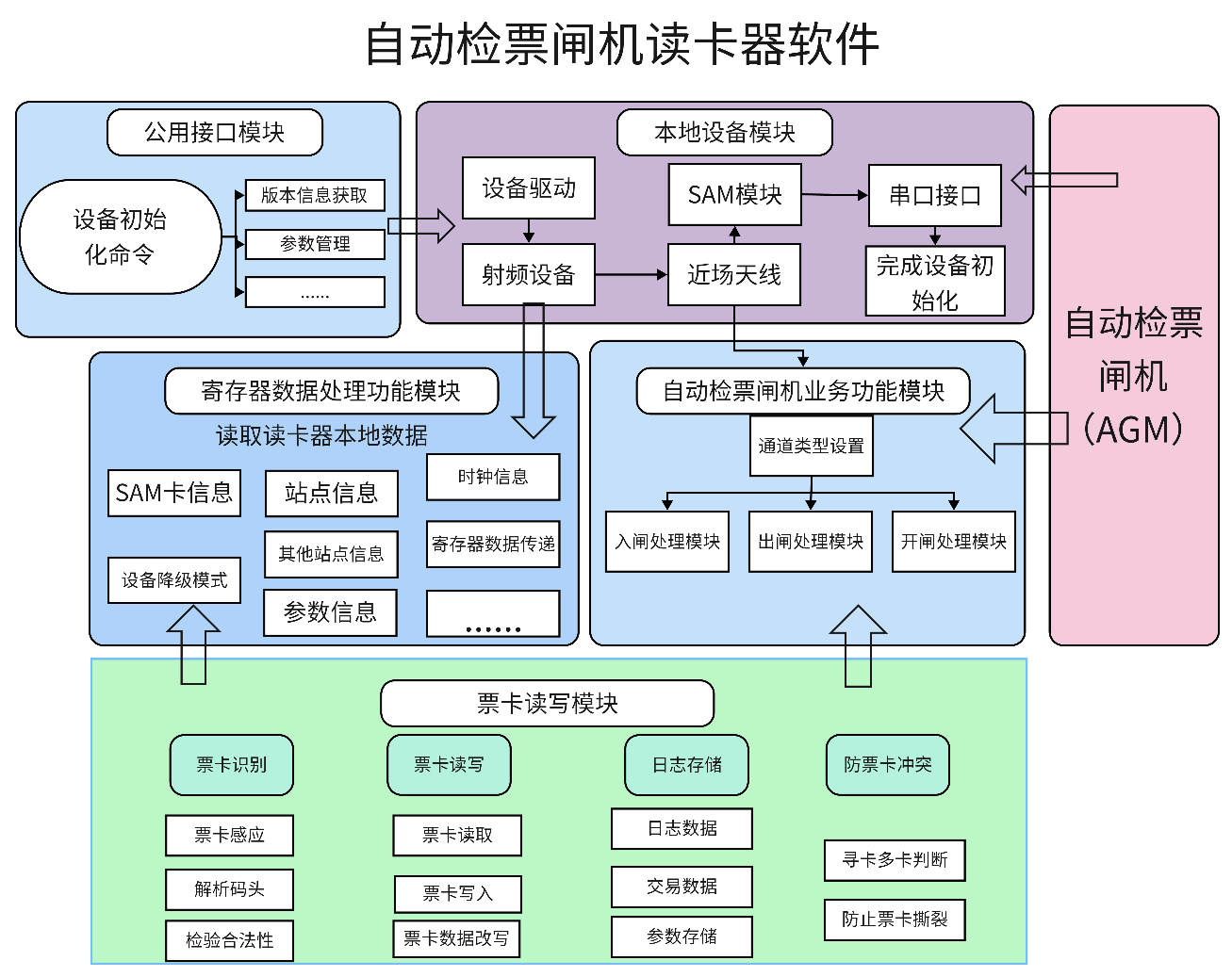
出站免检模式：进站闸机扇门处于关闭状态，乘客正常检票入站；出站闸机扇门处于全开状态，乘客出站不需检票，回收类车票不回收。

### 方案设计标准

### 方案总结

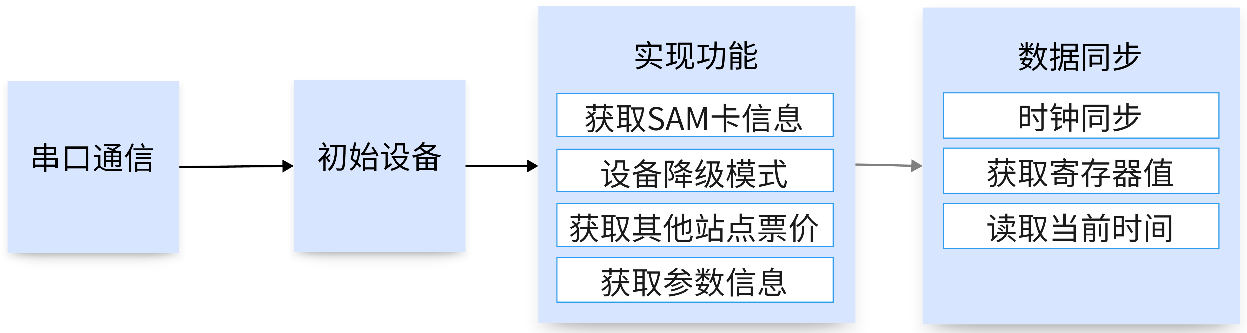
# 概要设计

## 读写器软件概要设计



本软件设计主要的三大模块：1、公用接口模块应用于读卡器与上位机进行通信、读卡器自初始化以及读取读卡器本地数据等功能。2、自动检票闸机业务功能模块应用于各种不同闸机的业务功能，包括闸机类型设置、出入闸机以及开闸处理等。3、票卡读写模块应用于地铁的验票流程，涵盖票卡感应、票卡识别、防票卡冲突等功能。

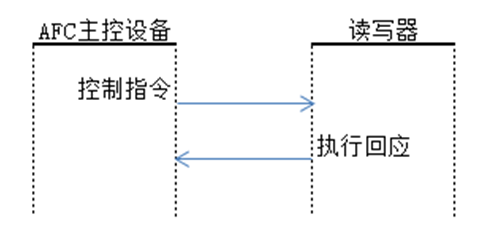
3.1.1 公用接口子模块设计



**1、串口通信**

UART串口最精简的连接是TTL电平三线连接UARTx\_TXD：用于发送数据，应连接到接收设备的UARTx\_RXD引脚上；UARTx\_RXD：用于接收数据，应连接到发送设备的UARTx\_TXD引脚上；GND：为双方提供一个相同的参考电平。

所有业务逻辑由非接触读卡器实现。当需要对读写器进行设置或读写卡片时，终端向读写器发送相应指令，由读写器返回操作结果。若终端未发起支付交易命令，读卡器应暂时禁止寻卡功能。除了从读卡器接收卡片信息外，终端还可能需要执行其他功能，因此必须确保报文流程不会显著的增加交易时间。

****

**2、设备初始化**

初始化读卡器的各硬件模块。根据接口传入的参数，加载设备所需参数，申请内部资源设备要调用读写器处理票卡前必须先执行此调用。设备在初始化之前应先调用‘Config\_Parameter’配置好设备必须的参数.

**3、版本信息获取**

读取读写器对应的版本信息。

**4、获取SAM卡信息**

读取读写器对应类型的SAM卡相关信息。

**5、设备降级模式**

降级模式是统称,它是指在出现列车运营故障，部分车站暂时中止运营服务时，暂停服务的车站需根据相关规定设置列车故障模式。在此模式下，对车票要进行特别处理。模式结束后，所有的检票机也要对车票进行处理。

当SLE（本地无线交换机）接收到SC下发的降级模式通知或降级模式控制命令后，SLE将通知或控制命令的内容填入结构pDegradeCmd后调用本接口，本接口被调用后，API不对pDegradeCmd信息做永久记录，API从内存释放后pDegradeCmd信息丢失，API被再次载入内存时，上次载入时的pDegradeCmd信息被忽略，除非上位机软件再次调用本接口。

上位机软件应永久记录pDegradeCmd相关信息直至该信息失效（当前时间超出该降级消息的敏感期），并在每次调用完初始化接口后调用本接口。

当有新的降级模式参数后，API将以降级模式参数为准，忽略之前所有调用本接口所产生的降级。

**6、获取其他站点到本站的票价**

获取当前时间其它站到本站的票价，做付费区无进站码更新时可能需要额外补款，可以通过本接口配合。

**7、获取参数信息**

读取读写器相应参数信息。

**8、时间同步**

同步读写器与上位机通信时钟。

**9、获取单个寄存器值**

读取当前设备固定地址的寄存器

**10、获取读卡器当前时间**

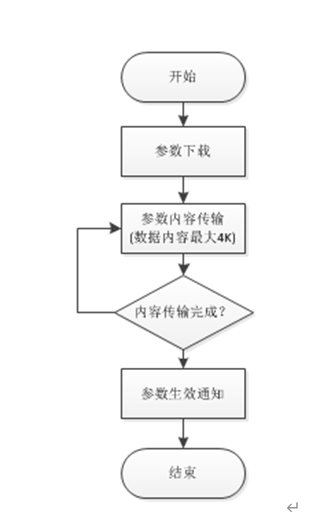
读取读卡器当前的时间

3.1.2运营参数模块设计

**1、进行参数下载**

当参数表有更新时，ECU上层应用程序将参数表文件下载到本地存储器中，并调用此接口函数通知API启用此参数文件。

参数下载流程:



参数下载生效通知

参数下载开始前和参数下载完成后调用。

参数内容传输

将参数文件分包下发到读写器，每次下发最大数据内容为4Kbyte。

2、获取参数信息

获取当前启用指定参数表的文件名，不包含全路径。

3.1.3 自动检票闸机业务功能模块设计



图3.2.1 一般AGM机业务功能

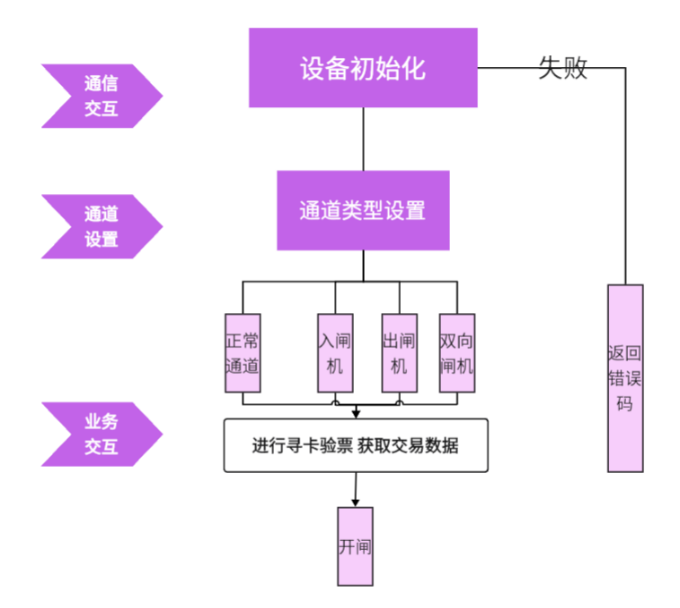


图3.2.2 读卡器与AGM闸机业务交互流程

**1、通道类型设置模块**

设定读写器当前所属通道类型，此函数只适用于闸机，闸机车票处理流程中读写器根据此通道类型去处理对应类型的车票。

上位机应用软件调用完读写器初始化接口后紧接着调用此接口函数，如不调用此接口函数，在入/出闸处理中做为正常通道处理。

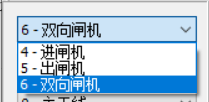


图3.2.3 AGM闸机通道类型

**2、入闸处理模块**

在入口闸机中调用此接口，成功进行交易后返回进闸交易数据，否则返回操作错误状态码。如果扩充保留字段为0x11,0x00，代表为电子车票业务，电子票入闸接口仅返回ITP验证电子票合法性的票卡结构，交易数据由开闸处理接口产生。

**3、出闸处理模块**

在出口闸机中调用此接口，根据应用情况指定主天线处理储值票或是副天线处理单程票，成功进行交易后返回出闸交易数据，否则返回操作错误状态码。如果扩充保留字段为0x11,0x00，代表为电子车票业务，电子票出闸接口仅返回ITP验证电子票合法性的票卡结构，交易数据由开闸处理接口产生。

**4、开闸处理模块**

当调用完入闸和出闸接口后，如果返回为电子票交易时，需要在调用此接口，获得开门指令以及交易数据。

3.1.4 票卡读写模块设计

票卡交易的最主要组成部分是验票过程，验票主要是读取车票内的发行信息，当前信息及历史交易信息。验票时AFC系统专用读写器读取乘客所持票卡的信息，根据卡内文件数据判断卡是否在系统内有效。在进站时，AFC系统专用读写器向交易文件中写入入站站信息和入站时间信息，单程票和储值票均可被AFC系统专用读写器处理。在出站时读写器读票卡交易文件中的入站时间信息，查看是否超程或超时；读取入站站信息计算乘车区间和票价，读智能卡的钱包值文件，并从中扣除相应的金额(或次数)。其次，结合票卡数据结构，研究交易过程的防撕裂操作。最后，考虑可能出现的各种情况，分析票卡交易的具体流程。

**1、票卡识别**

包括寻卡、解析码头、得到密钥、先解密二维码转成明文、校验票卡合法性、明文分析。

**2、票卡读写**

包括对票卡数据的访问和读写功能，实现票卡的读写。

**3、日志存储**

具备能够存储大量数据的功能（日志、交易数据、参数文件存储）。

**4、防票卡冲突**

具有防冲突机制（寻卡多卡判定）。

## 寄存器数据设计

### 寄存器数据存储

|  |  |
| --- | --- |
| 英文 | 说明 |
| ACC（AFC Clearing Center） | 清分中心系统 |
| LCC（Line Central Computer） | 线路中央计算机 |
| SC （Station Computer） | 车站计算机 |

(1)寄存器数据由车站设备产生，寄存器数值从设备投入使用之日起开始累计。

(2)寄存器数据采用四字节整型，金额单位为分，累计到最大值之后清零重新开始累计。

(3)SC定时（定时间隔由SC参数设定，默认为15分钟）向设备发送查询寄存器数据的指令，设备（仅包含闸机、BOM、TVM三类设备）以SOCKET消息报告当前的寄存器数据，或SC操作员查询时，设备报告当前寄存器数据。

(4)SC以SOCKET消息方式，向LCC转发所收到的寄存器数据消息，但不存储到本地数据库。

(5)LCC将收到的寄存器数据消息定时（定时间隔由LCC参数设定，默认为15分钟）打包成一个文件（按线路打包），以文件形式转发ACC，不存储到本地数据库。

(6)ACC将寄存器数据文件解包并存储到本地数据库。

AGM 审计数据的流向：AGM->SC->LCC->ACC。

### AGM机寄存器数据设计

1、AFC设备请求协议格式：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| AFC设备主控制器发起报文-命令报文 | | | | |
| 数据名称 | 长度 | 偏移 | 数据类型 | 说明 |
| 消息头部 | 1 | 0 | byte | AA（设备主程序发送） |
| 节点编码 | 1 | 1 | byte | 接收端地址码，用于一个端口接多个读写器  默认为0x01 |
| 报文序列号 | 1 | 2 | byte | 由上位机管理，0开始，每次通讯完加1，到0xFF后重新归零。 |
| 指令类码 | 1 | 3 | byte |  |
| 命令码 | 1 | 4 | byte |  |
| 参数1 | 1 | 5 | byte[] | 预留 |
| 参数2 | 1 | 6 | byte | bit7:重试标志位(0:新的请求,1:重试获取上次应答内容)  bit6-bit4预留  bit3-bit2:蜂鸣次数最多三次  bit1-bit0:天线模式0=主天线，1=副天线，2=第三天线 |
| 时间戳 | 7 | 7 | bcd[] | 主控设备当前时间,bcd码表示,YYYYMMDDHHNNSS |
| 数据长度 | 2 | 14 | word | 数据域长度，可以等于0；最大为4KByte |
| 数据域 | N | 16 |  | 控制指令相关的数据内容，可等于0 |
| 报文校验码 | 2 | 16＋N | word | 以上所有数据的CRC16计算值  Crc多项式0x1021，Crc预值0x00  例: 0x31 0x32 0x33 0x34 计算的结果为 0xD789 |

2、 读卡器应答协议格式

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 读写器回应报文-回应报文 | | | | |
| 数据元 | 长度 | 偏移 | 数据类型 | 说明 |
| 消息头部 | 1 | 0 | byte | BB（读写器发送） |
| 节点编码 | 1 | 1 | byte | 本设备地址码, 0x01 |
| 序列号 | 1 | 2 | byte | 填充当次处理指令收到的序列号 |
| 指令类码 | 1 | 3 | byte |  |
| 命令码 | 1 | 4 | byte |  |
| 长度 | 2 | 5 |  |  |
| 数据域 | N | 7 |  |  |
| 报文校验码 | 2 | 7＋N | word | 以上所有数据的CRC16计算值 |

3、协议格式中数据类型定义

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型 | 长度 | 字节顺序 | 例子 |
| hex | 1 | Intel | 内存地址值:{0x0A} 表示整数10 |
| byte | 1 | Intel | 内存地址值:{0x0A} 表示整数10 |
| bcd | 不定长 | Motorola | 内存地址:{0x20,0x15,0x10,0x26,0x16, 0x38,0x29} 表示整数2015年10月26日16时38分29秒 |
| word | 2 | Intel | 内存地址:{0x01,0x23} 表示整数0x2301(8961) |

说明:

Intel顺序表示为小端方式,即低位在前,高位在后

Motorola顺序为大端方式,即高位在前,低位在后

3、**SAM卡全局相关信息**

|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| keyidx | Unsigned char | 密钥索引 |
| len\_terminal\_id | Unsigned char | 终端号长度 |
| sam\_terminal\_id [8] | Unsigned char | 终端号 |
| len\_logical\_id | Unsigned char | 逻辑卡号长度 |
| offset\_logical | Unsigned char | 取逻辑卡号的偏移 |
| sam\_logical\_id [8] | Unsigned char | 逻辑卡号 |
| dsn | Unsigned Int |  |

4、错误提示信息结构

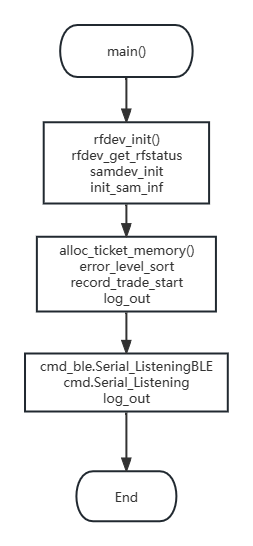
|  |  |  |
| --- | --- | --- |
| 字段名称 | 数据类型 | 说明 |
| wErrCode | Uint16\_t | 错误码 |
| bNoticeCode; | Uint8\_t | 关联提示码 |
| bRfu[2] | Uint8\_t | 扩充保留字段 |

# 详细设计

## 软件总模块详细设计

1、读卡器软件CSReader

Main函数：



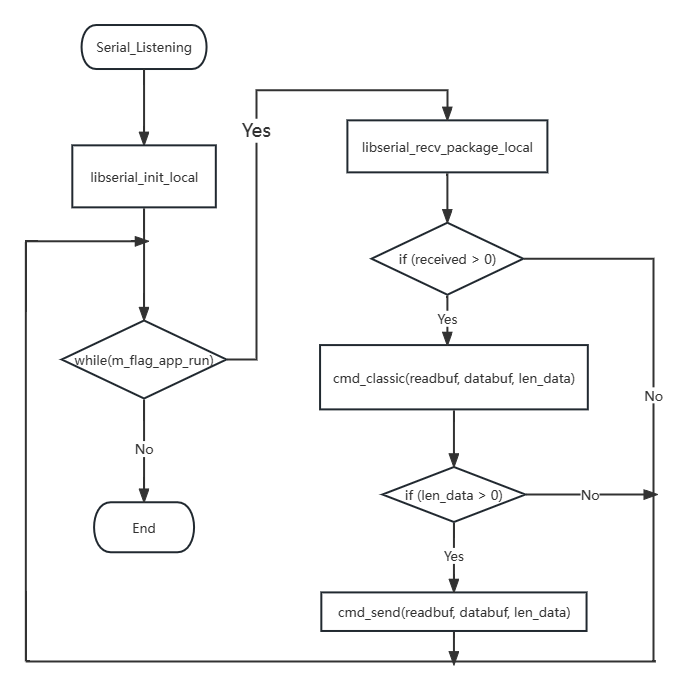
软件从main函数开始进入，第一个功能模块进行相关设备的初始化，涵盖主要函数rfdev\_inti()、rfdev\_get\_rfstatus、samdev\_init、init\_sam\_inf。

第二个功能模块为内存管理以及相关的日志输出。主要有函数alloc\_ticket\_memory()用于分配内存管理、error\_level\_sort record\_trade\_start、log\_out写日志， err\_code:当前调用错误码，lvl:当前指定级别，只影响本次，只要通过err\_code查找到的级别或者lvl\_appoint级别高于系统日志级别，就写日志。

第三个模块是对上位自动检票闸机（AGM机）报文命令的监听。主要由蓝牙和串口接口进行相关功能。

2、串口监听函数

int CmdSort::Serial\_Listening()：



cmd.Serial\_Listening为串口监听函数，首先是进行串口通信的设置。libserial\_init\_local为串口初始化函数，设置相关的波特率、数据位、校验码等。紧接着是对m\_flag\_app\_run的判断，若其大于0则进入while循环体，用于不断监听AGM机的报文命令，否则结束。

libserial\_recv\_package\_local为串口接收函数。进行AGM机报文命令的接收。判断接收值是否大于0，如大于0则进行相关报文命令的功能操作。

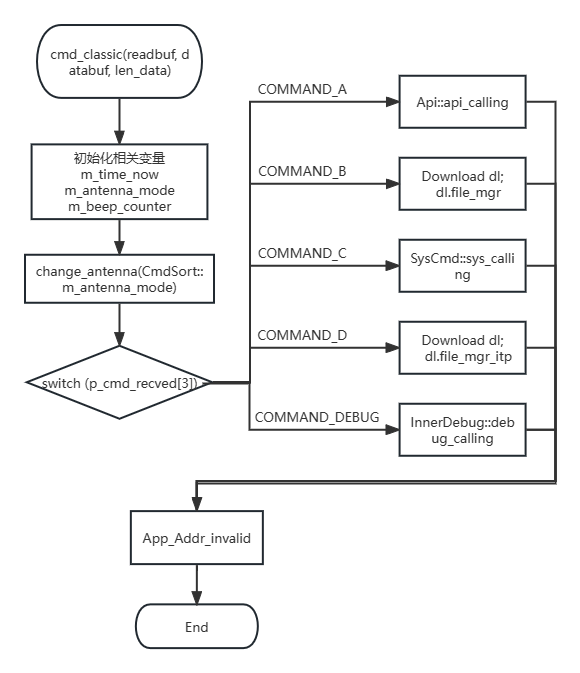
cmd\_classic(readbuf, databuf, len\_data)为报文命令功能操作函数，涵盖五种不同的功能模块。包括AGM机呼叫命令、文件管理、系统呼叫命令、参数下载、调试命令。

cmd\_send(readbuf, databuf, len\_data)为发送数据报文函数。判断返回数值len\_data是否大于0，若大于0则启用该数据发送函数进行读卡器报文命令的输出，返回至AGM机。若小于0则继续进行whlie循环体。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | | 数据类型 | 说明 | 详细内容 | |
| COM1 | | Char | 通信串口 | 软连接至(char \*)/dev/ttyS0 | |
| m\_flag\_app\_run | Bool | true | 真值 |
| readbuf | | Uint8\_t | 读写缓冲 | 初始值为0 | |
| databuf | | Uint8\_t | 数据缓存 | 初始值为0 | |
| len\_data | | Uint16\_t | 发送的数据 | 初始值为0 | |
| received | | Int | 串口通信返回值 | >0:跳转报文命令  ≤0：继续监听串口 | |

3、功能操作函数

void CmdSort::cmd\_classic(uint8\_t \* p\_cmd\_recved, uint8\_t \* p\_data\_to\_send, uint16\_t& lentosend)：



CmdSort::cmd\_classic函数为功能操作函数。首先要进行变量的初始化和变量的内存分配。然后开启天线模块。change\_antenna(CmdSort::m\_antenna\_mode)为其操作函数。

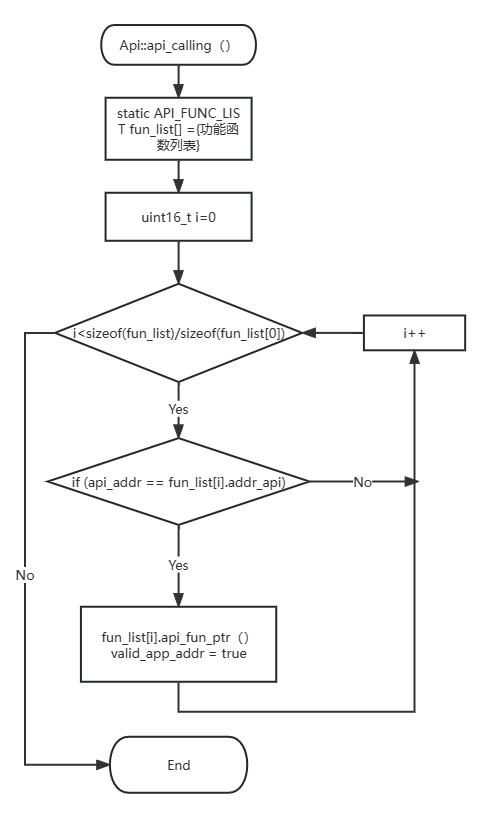
接着通过传入的readbuf数值，判断p\_cmd\_recved[3]为哪种情况。通过switch语句进行不同功能模块的选择。若是COMMAND\_A，则进行函数Api::api\_calling（），作用是完成AGM机的报文命令。若是COMMAND\_B，则是dl.file\_mgr，作用是进行文件管理和参数下载。若是COMMAND\_C，则是SysCmd::sys\_calling，完成系统呼叫命令。若是COMMAND\_D，则是dl.file\_mgr\_itp，进行参数下发通知。若是COMMAND\_DEBUG，则是InnerDebug::debug\_calling，用于人工调试命令。

最后进行地址非法错误码返回函数Api::App\_Addr\_invalid。若app\_valid为fales则进行错误码返回。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 | |
| m\_time\_now | | Uint8\_t | 当前时间 | 从p\_cmd\_recved获得数值 |
| m\_antenna\_mode | | Uint8\_t | 天线模块 |  |
| m\_beep\_counter | | Uint8\_t | 蜂鸣器计数 |  |
| p\_cmd\_recved[] | | Uint8\_t | readbuf | AGM机发送的报文 |
| COMMAND\_A | | #define | (uint8\_t)1 |  |
| COMMAND\_B | | #define | (uint8\_t)2 |  |
| COMMAND\_C | | #define | (uint8\_t)3 |  |
| COMMAND\_D | | #define | (uint8\_t)4 |  |
| COMMAND\_DEBUG | | #define | (uint8\_t)0x10 |  |
| app\_valid | | Bool | 初始值为true |  |
| lentosend | | Uint16\_t& | Switch后功能返回值 | 执行不同情况功能后的返回值 |
| wErrCode | | Uint16\_t | 错误码 |  |

4、AGM机报文命令操作函数

void Api::api\_calling(uint8\_t api\_addr, uint8\_t \* param\_stack, uint8\_t \* data\_to\_send, uint16\_t& len\_data, bool& valid\_app\_addr)：



Api::api\_calling（）为AGM机报文命令功能操作函数。首先要进行初始变量的赋值。fun\_list[ ]为功能函数列表数组。详细的函数列表如下表所示。

接着进入for循环体。设置初始变量i为0，判断i是否小于fun\_list[ ]功能函数列表中封装的函数个数。若大于或等于则直接结束函数运行。若其小于封装的列表函数个数，则进行对应的函数功能操作。进行对应功能函数操作还需判断fun\_list[ i]中的addr\_api是否等于从AGM机接收到的报文命令中的p\_cmd\_recved[4]，若等于则启用对应的封装列表中的功能函数，并且进行i++，回到for循环开头。若addr\_api不等于p\_cmd\_recved[4]，则直接进行i++回到for循环开头。

以此循环，不断检索封装列表中的功能函数，直至符合AGM机的报文命令指示后，启用对应的功能函数。

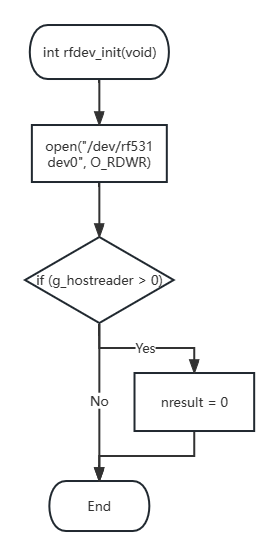
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 | |
| fun\_list[] | | Typedef struct | 结构体类型 | 存储了功能函数的编号 |
| i | | Uint16\_t | 变量 | 用于循环检索相应功能函数 |
| fun\_list[i].addr\_api | | Uint8\_t | 编号 | 列表中的功能函数编号 |
| api\_addr | | Uint16\_t | readbuf | AGM机发送的报文 |
| valid\_app\_addr | | bool | True | 函数运行成功返回true值 |

|  |  |  |
| --- | --- | --- |
| i（fun\_list[i]）列表函数编号 | 功能函数名称 | 说明 |
| 0 | Common\_Initialize\_Device | 设备环境初始化 |
| 1 | Common\_GetVersion | 获取版本信息 |
| 2 | Common\_GetSamInfo | 获取SAM卡信息 |
| 3 | Common\_SetDegradeMode | 设备降级模式 |
| 4 | GetFare | 获取其他站点到本站的票价 |
| 5 | Common\_GetParamInfo | 获取参数信息 |
| 6 | Get\_Reg\_Value | 获取单个寄存器值 |
| 7 | Get\_Reg\_Inf | 获取所有寄存器值 |
| 8 | Reset\_Reg\_Inf | 重置寄存器 |
| 9 | Common\_LastTrade | 获取上次通讯数据 |
| 10 | Common\_TimeSynchronization | 时间同步 |
| 11 | Common\_GetReaderTime | 获取当前读卡器时间 |
| 12 |  |  |
| 13 | Gate\_AisleModel | 通道类型设置 |
| 14 | Gate\_EntryFlow | 入闸处理 |
| 15 | Gate\_ExitFlow | 出闸处理 |

**4.2 初始模块详细设计**

1、射频设备初始化函数

rfdev\_init()：

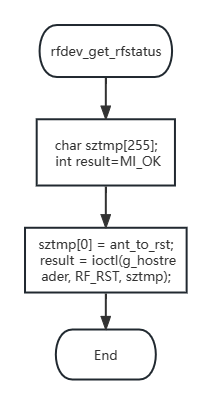


初始化射频设备首先定义nresult值，然后进行射频设备rf531的打开进而判断g\_hostreader值。若返回值为0则射频设备打开成功，将nresult值置为0。若返回值为-1则射频设备打开失败， nresult值不变。初始化射频设备函数结束。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| nresult | Int | 初始化射频设备返回值 | 0=成功  -1=打开设备句柄出错 |
| g\_hostreader | Int | 打开射频设备返回值 | 初始值：-1  打开成功：>0 |

2、天线设备初始化函数

rfdev\_get\_rfstatus：

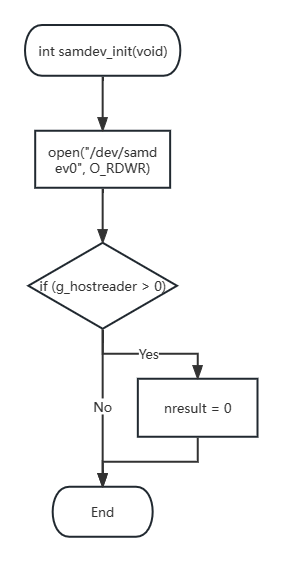


初始化天线设备，定义sztmp数组和result，进而赋值sztmp数组首位为1，然后执行ioctl函数，用于设备驱动程序中对设备的I/O通道进行管理对设备的一些特性进行控制。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| sztmp | Unsigned char | 初始化射频设备返回值 | 0=成功  -1=打开设备句柄出错 |
| result | Int | I/O设备管理及控制返回值 | 初始值为0  0=没有天线  1=RF1天线有效  2=RF2天线有效  3=RF1和RF2天线有效 |
| MI\_OK | #define | 0 |  |
| RF\_RST | #define | 24 |  |
| ant\_to\_rst | Unsigned char | 1 |  |

3、Sam卡设备初始化

samdev\_init()：



初始化SAM卡设备首先定义nresult值，然后进行SAM卡设备的打开进而判断g\_hostreader值。若返回值为0则SAM卡设备打开成功，将nresult值置为0。若返回值为-1则SAM卡设备打开失败， nresult值不变。初始化SAM卡设备函数结束。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| nresult | Int | 初始化SAM卡设备返回值 | 0=成功  -1=打开设备句柄出错 |
| g\_hostreader | Int | 打开SAM卡设备返回值 | 初始值：-1  打开成功：>0 |

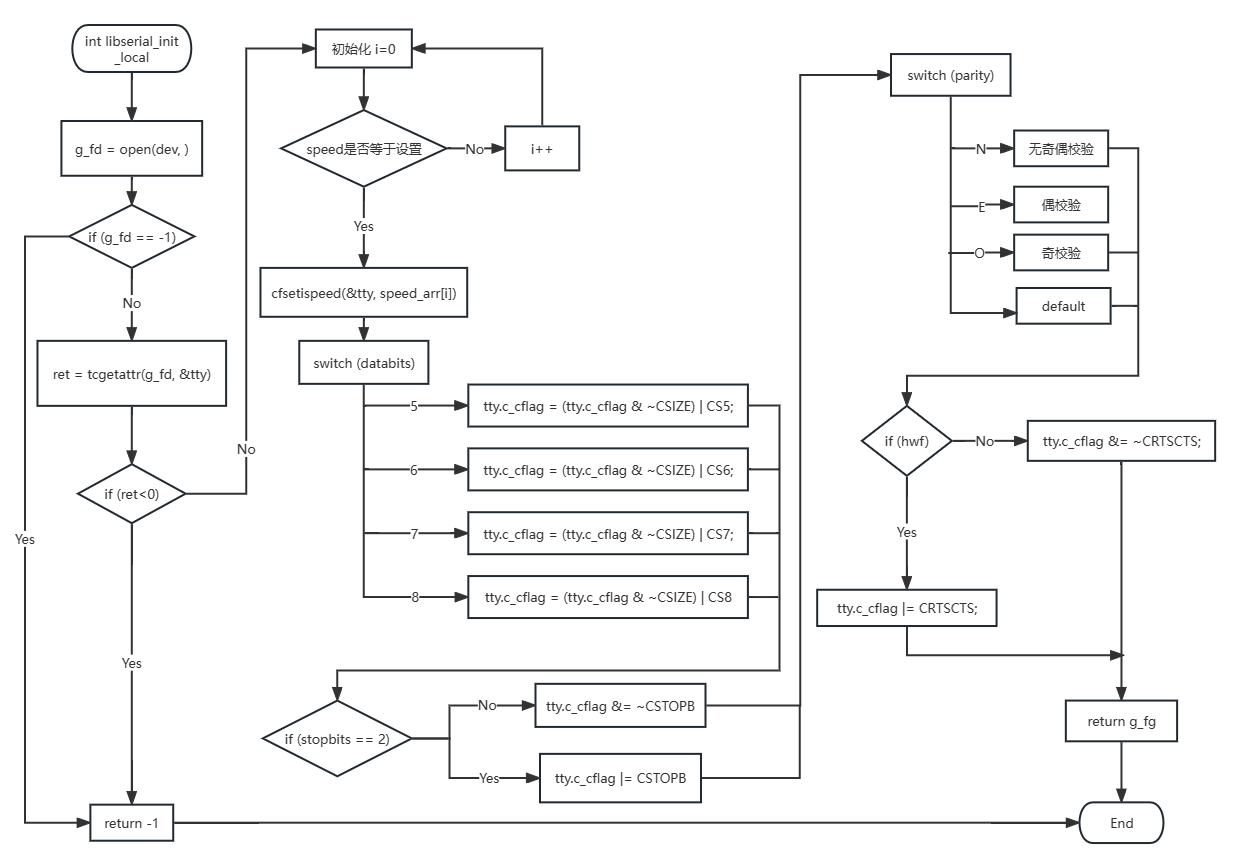
**4.3公用接口模块详细设计**

4.2.1串口通信编程

1、串口（UART）设置函数

在Linux系统中，操作设备统一接口就是：open/ioctl/read/write。对于UART，又在ioctl之上封装了很多函数，主要是用来设置行规程。所以对于UART编程的就是设置行规程，比如波特率、数据位、停止位、检验位、RAW模式、一有数据就返回。行规程的参数用结构体来表示。

int libserial\_init\_local(char \*dev, int speed, int parity, int databits, int stopbits, int hwf, int swf)：



（1）进入串口设置函数：首先是g\_fd = open(dev, O\_RDWR | O\_NOCTTY)函数。该函数作用是打开串口设备，并设置为可读可写以及非控制台状态。然后判断g\_fd是否等于-1，若为-1则是打开设备失败串口函数返回-1并结束。若非-1则继续下一步。tcgetattr函数用于获取与终端相关的参数。参数fd为终端的 [文件描述符](http://baike.baidu.com/view/1303430.htm)，返回的结果保存在termios [结构体](http://baike.baidu.com/view/204974.htm)中，如果参数返回值小于0则串口函数返回-1并结束。若非小于0则进行波特率的设置。

（2）设置波特率：要约定好传输速率（每一秒传输的数据位数，即波特率），一般选择9600,19200,115200等。进行for循环，初始化变量i，若i不等于相应的输入波特率，则i++，直到i等于需要的波特率。然后用cfsetispeed函数，将串口的波特率设置为当前i值。

（3）设置数据位：获取databits的值，为5、6、7或者8，并用tty.c\_cflag = (tty.c\_cflag & ~CSIZE) |databits函数设置好串口的数据位。数据位:可以有5、6、7或8位的数据，一般我们是按字节（8位）传输数据，发送方一位一位的改变数据线上的状态（高电平或低电平）将它们发送出去，传输数据时先传最低位，最后传送最高位。字符’A’的8位二进制字符是01000001，先发送最低位bit0，其值为1；再发送bit1，其值为0，如此继续；最后发送最高位bit7，其值为0

（4）设置停止位：停止位停止位（逻辑1）用来表示当前数据传输完毕。停止位的长度有三种：1位，1.5位，2位，这里选择2位。判断stopbits的值是否等于2，不为2则进行tty.c\_cflag &= ~CSTOPB用于停止位的设置。为2则进行一下步设置奇偶校验。

（5）设置奇偶校验：使用了奇偶校验功能，有效数据位发送完毕后，还要发送1个校验位（奇偶校验位）。有两种校验方法：奇校验，偶校验-------数据位连同校验位中，“1”的数目等于奇数或偶数。奇偶校验只能检错，不能纠错的。而且只能检测1位误码，检测出有错后只能要求重发，没法纠正的。判断parity值，为N则无奇偶校验，为O则是奇校验，为E则是偶校验。其余情况均非法结束。

（6）设置软流控制：最后判断hwf的值，如为硬流控制，则进行tty.c\_cflag |= CRTSCTS设置为软流。

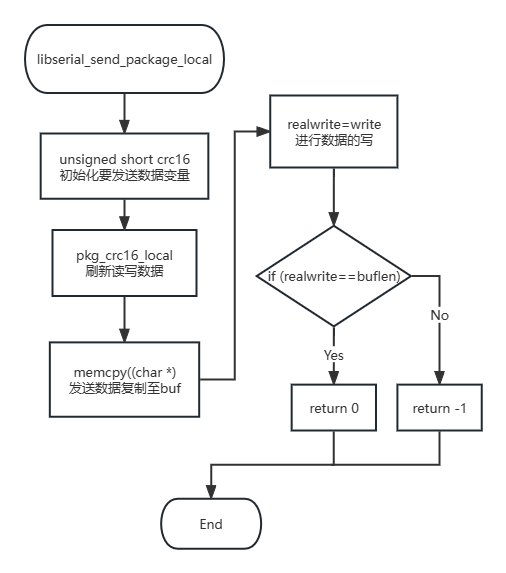
最后串口设置函数结束并返回g\_fg的值。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| dev | Char | 串口设备名 | 串口1是"/dev/ttyS0" |
| g\_fd | Bool | 设备打开标志 |  |
| ret | Bool | 设备参数返回值 | 参数返回值大于0则成功 |
| speed | Int | 串口波特率 | 可以是230400, 115200, 57600, 38400, 19200, 9600 |
| parity | Int | 奇偶校验 | 值为'N','E','O','S' |
| databits | Int | 数据位 | 值是5、6、7或者8 |
| stopbits | Int | 停止位 | 值是1或者2 |
| hwf | Int | 硬件流控制 | 1为打开，0为关闭 |
| swf | Int | 软件流控制 | 1为打开，0为关闭 |

2、串口数据发函数

发数据格式详见3.2.2AGM寄存器数据设计：读卡器应答数据报文表格

int libserial\_send\_package\_local(uchar \*buf, const int buflen);



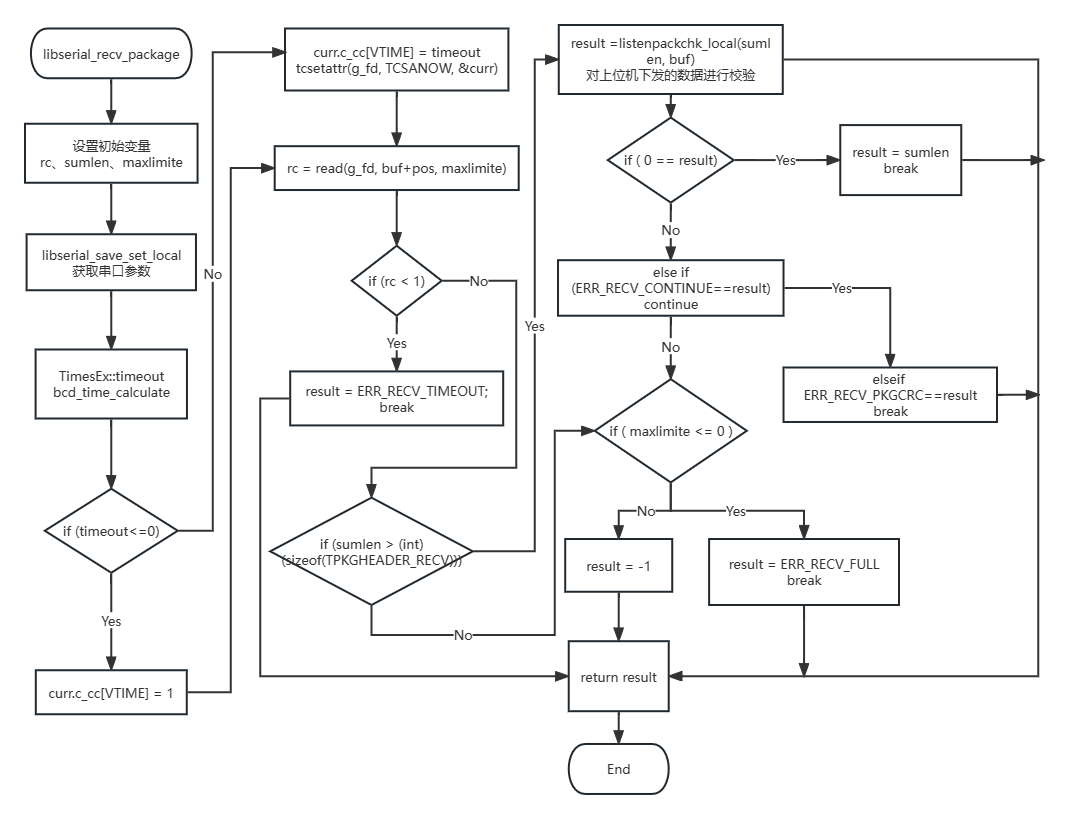
串口发送数据函数，首先初始化发送的数据变量crc16，接着进行读写数据的刷新。然后用memcpy函数麻将传入串口发送函数的参数即crc16内容复制至要发送的buf当中。再之后进行realwrite=write(g\_fd, buf, buflen)函数，将buf的内容写至通信串口。最后，对write函数值进行判断，如果realwrite==buflen，返回值等于相对于的buf长度则串口发送数据函数返回0并结束运行。若其不等，则串口发送函数返回-1并结束。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| crc16 | Unsigned short | 发送值 |  |
| buf | Bool | 发送缓冲 |  |
| buflen | Int | 数据长度 |  |
| realwrite | Int | write函数返回值 | 0=发送完成，-1=发送失败 |

3、串口收数据

收数据格式详见3.2.2AGM寄存器数据设计：AGM机发送数据报文表格

libserial\_recv\_package\_local(int maxrecv, int timeout, uchar \*buf)



（1）串口收发函数libserial\_recv\_package\_local，首先进行相关变量的初始化。然后启用函数libserial\_save\_set\_local，用于获取设备的相关参数。接着判断是否超时。进行bcd\_time\_calculate函数，用于BCD编码的时间加减。计算好接收时间后，返回timeout判断是否超时。若其小于等于0，进行curr.c\_cc[VTIME]定义等待的时间为100ms，若其大于0，则进行curr.c\_cc[VTIME] = timeout定义等待的时间为timeout的值。

（2）定义完等待时间后，进行read函数，对上位机字符进行接收。并返回rc的值，若rc小于1，则判断为超时。result = ERR\_RECV\_TIMEOUT，收数据函数返回result值并结束。若rc大于1则进行下一步。

（3）先判断sumlen 是否大于 (sizeof(TPKGHEADER\_RECV)，若大于则直接进行数据长度的判断，判断maxlimite 是否小于等于0 ，若其小于等于0则result = ERR\_RECV\_FULL，返回result，并结束收数据函数。若maxlimite大于0，则result = -1，并结束收数据函数。若sumlen小于 (sizeof(TPKGHEADER\_RECV)则继续下一步。

（4）下一步进行函数listenpackchk\_local，其功能是对上位机下发的数据进行校验，当数据包合法时，对数据内容进行处理并回传。回传值有以下四种情况0：数据有效，1：数据包头错，-2 ：长度不足，-3：校验错。然后对回传值进行判断，如果0 == result，则result = sumlen，数据有效，收数据函数返回result值并结束。如果ERR\_RECV\_CONTINUE==result，则判断maxlimite 是否小于等于0 ，若其小于等于0则result = ERR\_RECV\_FULL，返回result，并结束收数据函数。若maxlimite大于0，则result = -1，并结束收数据函数。如果ERR\_RECV\_PKGCRC==result，则报文校验码出错，收数据函数返回result值并结束。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| maxrecv | Uint8\_t | 接收缓冲大小 |  |
| timeout | Bool | 超时时间 | 超时时间，单位为100ms |
| buf | Bool | 接收缓冲 |  |
| rc | Int | 接收字符 | 判断接收值，<1是超时 |
| maxlimite | Int |  | 最大接收长度，防止缓冲溢出 |
| TPKGHEADER\_RECV | struct | 报文接收结构 |  |
| ERR\_RECV\_TIMEOUT | #define | 接收超时 | -1 |
| ERR\_RECV\_FULL | #define | 数据接收缓冲区满 | -2 |
| ERR\_RECV\_PKGCRC | #define | 数据校验错 | -3 |
| ERR\_RECV\_CONTINUE | #define | 接收未完成，继续 | -6 |
| result | Int | 收数据函数返回值 | 0=执行无返回  -1=接收超时  -2=接收缓冲区满  -3=crc校验错  >0=有正确的数据报文件，返回整个报文长度 |

4.2.2 设备初始化

{0, Common\_Initialize\_Device},//设备环境初始化

读写器在进行读写操作前必须进行相应的设备初始化操作，只有正确地进行了设备初始化操作后，读写器才能够分别进行TVM入闸出闸处理、BOM型读写器票务处理等，跳过此步骤直接进行票卡处理会使得读写器无法匹配到相对应的票卡信息，使得返回码不为零，出现错误。

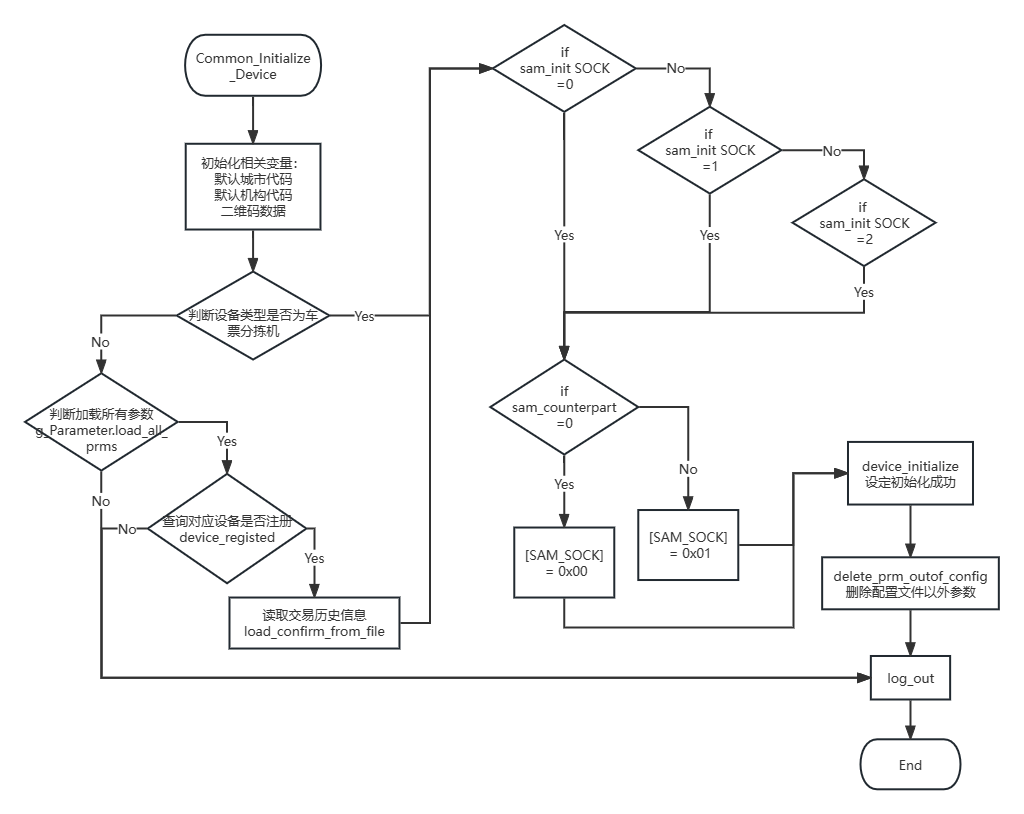
设备环境初始化根据接口传入的参数，加载设备所需参数，申请内部资源。设备要调用读写器处理票卡前必须先执行此调用。设备在初始化之前应先调用‘Config\_Parameter’配置好设备必须的参数，这些参数包括：

|  |  |
| --- | --- |
|  | 参数名称 |
| 1 | 设备控制参数 |
| 2 | 车站代码表 |
| 3 | SAM卡对照表 |
| 4 | 换乘车站代码表 |
| 5 | 票卡参数 |
| 7 | 票价参数 |
| 8 | 降级模式使用记录 |
| 9 | 地铁单个黑名单 |
| 10 | 地铁黑名单段 |
| 11 | 一卡通单个黑名单 |
| 12 | 一卡通黑名单段 |
| 13 | 互联网票单个黑名单 |
| 14 | 互联网票黑名单段 |
| 15 | 行政罚金参数 |
| 16 | 乘次票专用参数 |
| 17 | 闸机专用通道参数 |
| 18 | 二维码扫描器参数 |
| 19 | 蓝牙模块参数 |

其中第17个参数为读写器专用

1、初始化设备函数

void Api::Common\_Initialize\_Device(uint8\_t \* param\_stack, uint8\_t \* data\_to\_send, uint16\_t& len\_data)



Common\_Initialize\_Device为设备环境初始化函数。进入该函数，首先初始化相关变量。包括默认的城市代码、默认的机构代码和二维码数据等。接着判断设备类型，如果上位机是“车票分拣机”，则直接进行SAM卡的初始化。如若不是，则进行参数加载函数g\_Parameter.load\_all\_prms。如若参数加载出错，则直接进行日志的输出并结束初始化设备函数。如若参数加载成功，接着进行设备注册信息的查验，启动代码函数device\_registed。如若不通过设备注册信息的检验，则进行日志输出并结束初始化设备函数进行。如若通过设备注册信息的检验，进行函数load\_confirm\_from\_file用于读取交易的历史信息。

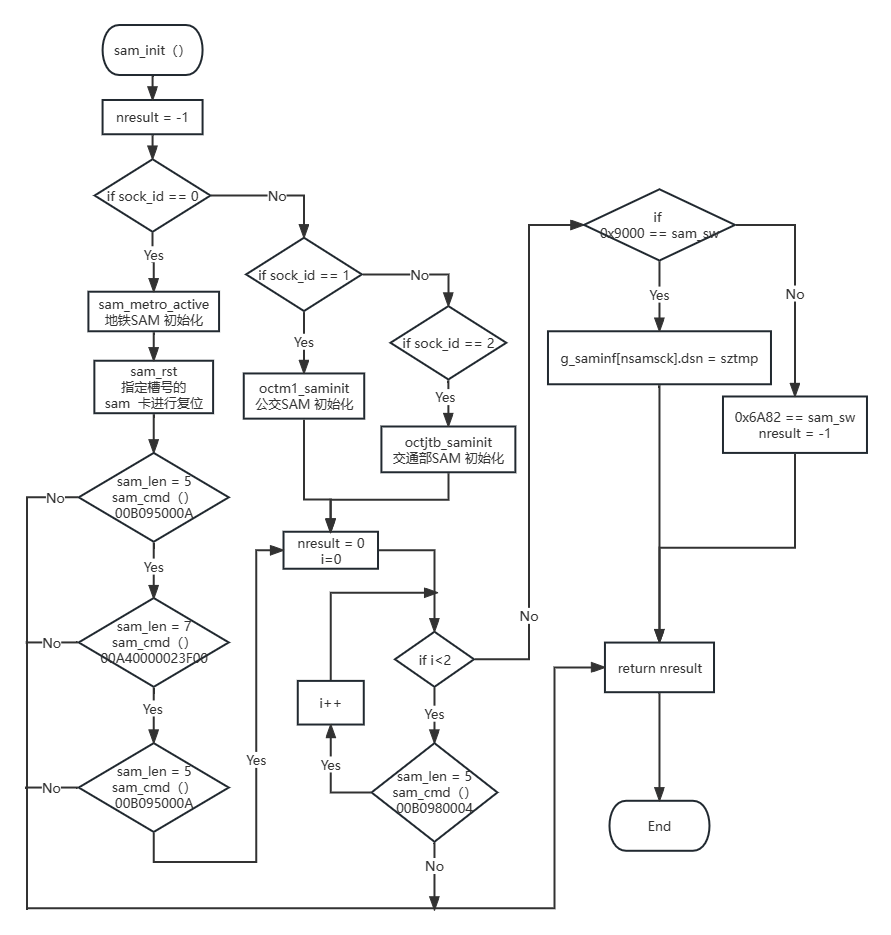
在完成相关参数的加载后，进行SAM卡的初始化。首先判断卡槽的槽号，包括0、1、2号卡槽，接着进行相应的sam init函数，完成SAM卡的初始化。待其初始化完成之后，进行SAM卡与设备注册信息的匹配，函数sam\_counterpart作用于此。若匹配，则SAM卡状态返回值为0x00，如若不匹配，则sam卡槽返回值为0x01。

判断完SAM卡槽状态后，进行函数device\_initialize设定参数为初始化成功，然后进行函数delete\_file\_overdue删除除配置文件外的参数配置，最后进行日志输出并结束初始化函数。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| current\_station\_id | Uint16\_t | 当前车站代码 |  |
| current\_city\_id | Uint16\_t | 当前城市代码 |  |
| bLocalInstitutionCode | Uint8\_t | 受理机构代码 |  |
| device\_type | Uint8\_t | 设备类型码 | 09为车票分拣机  4为双通道闸机  5为入通道闸机  6为出通道闸机 |
| SAM\_SOCK\_i | #define | SAM卡卡槽号 | 0、1、2 |
| sam\_status | Uint8\_t | SAM卡与设备注册信息状态 | 0x00 为正确  0x01 为错误 |
| device\_init\_flag | Bool | 初始成功标志 |  |
| err\_code |  | 当前调用错误码 | err\_code查找到的级别高于系统日志级别，就写日志 |

2、SAM卡初始化函数

uint16\_t sam\_init(int sock\_id, char \* p\_sam\_id, char \* p\_tml\_id)



进入初始化SAM卡函数sam\_init（），首先初始化变量nreuslt=-1，用于函数结束时判断是否初始化成功。然后判断此时的要初始化的SAM卡槽是哪种SAM卡，如果卡槽号为1，则是公交SAM卡的初始化函数。如果卡槽号为2，则是交通部SAM卡的初始化函数。本论文的设计实现为地铁AFC设备读卡器。因此着重于地铁SAM卡的初始化。当卡槽号为0时，进行地铁SAM卡的初始化函数sam\_metro\_active（）。

该函数首先进行sam卡相关信息的打印，接着启用函数sam\_cmd（），用于发送数据报文命令给SAM卡进行数据的校验。执行三种不同的数据校验，如果出错则直接返回nresult=-1，初始化失败，结束函数运行。如若正确则nresult赋值为0，进入for循环，再进行两次公共的SAM卡数据校验。检验失败则输出返回值，并结束函数进行。检验成功则判断流水号sam\_sw是否正确，当其值为0x9000时则是正确，输出SAM卡全局相关信息，包括：密钥索引、终端号长度、终端号逻辑卡号长度、取逻辑卡号的偏移、逻辑卡号等。若其的值不为0x9000，则返回错码6A84。结束判断sam\_sw的值后，返回nresult的值并结束SAM卡初始化函数的运行。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| current\_station\_id | Uint16\_t | 当前车站代码 |  |
| current\_city\_id | Uint16\_t | 当前城市代码 |  |
| bLocalInstitutionCode | Uint8\_t | 受理机构代码 |  |
| device\_type | Uint8\_t | 设备类型码 | 09为车票分拣机  4为双通道闸机  5为入通道闸机  6为出通道闸机 |

4.2.3 版本信息获取

{1, Common\_GetVersion},//获取版本信息

4.2.4 获取SAM卡信息

{2, Common\_GetSamInfo},//获取SAM卡信息

4.2.5 设备降级模式

{3, Common\_SetDegradeMode},//降级模式消息

4.2.6 获取票价

{4, GetFare},//获取到本站的票价

4.2.7 参数管理

{5, Common\_GetParamInfo},//获取参数信息

4.2.8 时间同步

{10, Common\_TimeSynchronization},//时钟同步

4.2.9 获取寄存器值

{6, Get\_Reg\_Value},//获取单个寄存器值

{7, Get\_Reg\_Inf},//获取所有寄存器值

{8, Reset\_Reg\_Inf},//重置寄存器

{9, Common\_LastTrade},//获取上次通讯数据

4.2.10 获取读卡器当前信息

{11, Common\_GetReaderTime},//获取读写器当前时间

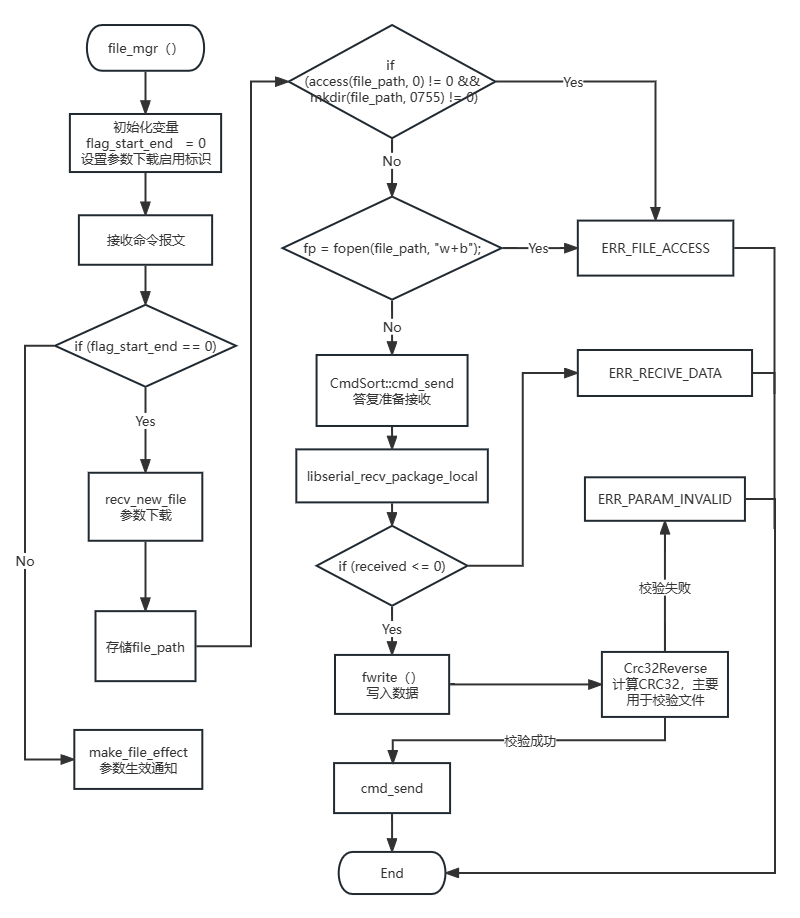
**4.4运营参数模块详细设计**

从软件总模块功能函数操作可知运营参数操作为case COMMAND\_B:

Download dl; dl.file\_mgr(p\_cmd\_recved, p\_data\_to\_send, lentosend);

4.4.1参数下载

recv\_new\_file(p\_msg\_file, file\_name, file\_size, file\_crc, p\_data\_send, len\_send);



dl.file\_mgr该函数用于参数管理，进入参数管理函数，首先先初始化参数下载标识为0，然后依照接收的上位机命令报文进入判断。如果加载报文命令后，标识仍为则启动参数下载函数recv\_new\_file()，否则启动参数生效函数make\_file\_effect()。

进入recv\_new\_file先一个指定的路径下创建一个临时文件夹。具体来说：running\_directory() 返回当前应用程序运行的目录（即当前路径）。表示临时文件夹的名称。使用 sprintf() 函数将其与宏定义NAME\_TEMP\_FOLDER结合起来，形成一个完整的路径字符串。存储到file\_path中。

然后判断指定的文件夹路径是否存在。：access(file\_path, 0)函数检查 file\_path 变量表示的路径是否存在，并且当前用户是否有读取权限。如果返回值为0，表示该路径存在且可读取；如果返回值不为0，说明路径不存在或者没有读取权限。如果路径不存在，则使用 mkdir() 函数来创建该路径。mkdir(file\_path, 0755) 中的 0755 表示权限控制，其中第一个数字7表示owner（文件所有者）的权限，第二个数字5表示group的权限，第三个数字5表示其他人的权限。其中的数字可以根据具体需求进行调整。该代码段整体的作用是：如果指定的路径不存在，就创建该路径。如果创建文件夹失败则返回错误码ERR\_FILE\_ACCESS。

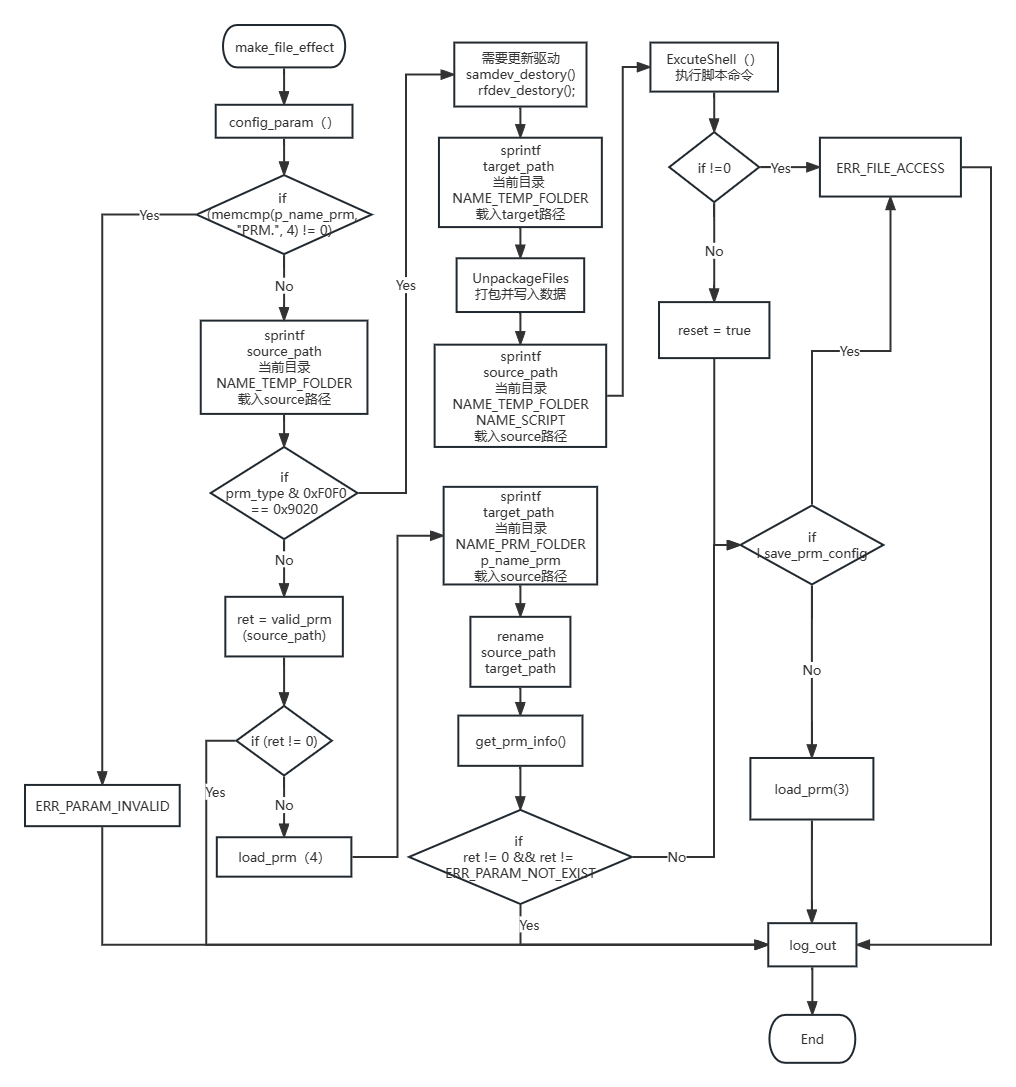
接着使用 fopen() 函数打开file\_path 的文件，以 w+b 的模式打开，既可以写入，也可以读取二进制数据。如果打开文件失败，fopen() 函数将返回 NULL，程序检查文件指针是否为 NULL，如果是，则表示打开文件失败。返回错误码ERR\_FILE\_ACCESS。

打开文件成功之后，读卡器返回应答报文，即 CmdSort::cmd\_send()函数用于答复准备接收。再之后进行数据的写入，写入完成之后会进行一个CRC校验。如若校验失败，则返回ERR\_PARAM\_INVALID。如果校验成功则返回写入成功的应答报文。结束参数下载函数，等待上位机的下一命令。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| flag\_start\_end | uint8\_t | 参数下载启用标识 | 0=参数下载通知  1=参数生效通知 |
| file\_path | char | 下载路径字符串 | 由当前目录与Tmp拼接 |
| NAME\_TEMP\_FOLDER | char \* | Tmp |  |
| ERR\_FILE\_ACCESS | #define |  | 文件错误 |
| fp | typedef struct | 文件打开标识 |  |
| total\_recv | uint32\_t | 下载文件大小 |  |
| crc\_calc | uint32\_t | CRC校验成功与否标识 | 初始值为：0xFFFFFFFF |

4.4.2 参数生效通知

make\_file\_effect(p\_msg\_file, file\_name, file\_size, file\_crc, p\_data\_send, len\_send);



图参数生效函数流程图

当参数下载启用标识为1时，进入make\_file\_effect（）参数生效函数，启用函数config\_param（）进行参数配置。首先判断下发的参数文件是否合法。若与字段"PRM."

不符合，则是返回结果ERR\_PARAM\_INVALID（参数文件不合法）。若是通过合法判断。进获取source的字符串路径，其由当“前路径+ NAME\_TEMP\_FOLDER+部分接收数据”组成。

接着用参数类型码和0xF0F0进行与运算，运算结果与0x9200若等于，则存在需要更新驱动的情况。进行函数samdev\_destory()和rfdev\_destory()用于相关驱动的更新。更新完驱动进行target字符串路径的更新载入，其由“当前路径+ NAME\_TEMP\_FOLDER”组成。然后启用函数:UnpackageFiles（）用于打包并写入参数数据。完成该函数后，获取source的字符串路径，由“当前路径+NAME\_TEMP\_FOLDER+ NAME\_SCRIPT”组成。再然后用函数ExcuteShell命令Linux开发板执行source路径下脚本命令。若执行失败，返回ERR\_FILE\_ACCESS，执行成功则结束该模块。

如若参数类型码和0xF0F0进行与运算，运算结果与0x9200若不等于，则是其他参数类型处理。启用函数valid\_prm（）用于检查参数的合法性。若参数不合法，则输出日志，结束参数生效函数运行。参数合法则进行函数load\_prm（）加载当个参数。加载失败同样输出日志并结束参数生效函数运行。加载成功则获取target的字符串类型路径，由“当前路径+NAME\_PRM\_FOLDER+数据接收部分”组成。接着用函数rename（），将target的字符串赋给source字符串。然后进行函数get\_prm\_info（）获取参数信息。获取失败返回ERR\_PARAM\_NOT\_EXIST，成功则结束该模块。

在结束以上任一功能模块后，进行参数配置的更改即函数save\_prm\_config（）。更改失败则返回ERR\_FILE\_ACCESS。更改成功则再次运行函数load\_prm（）加载单个参数。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| p\_name\_prm | char \* | 接收的部分报文 |  |
| ERR\_PARAM\_INVALID | #define | 参数文件不合法 | 0x0101 |
| ERR\_FILE\_ACCESS | #define | 文件错误 | 0x0102 |
| ERR\_PARAM\_NOT\_EXIST | #define | 参数文件不存在 | 0x0100 |
| NAME\_TEMP\_FOLDER | char \* | Tmp |  |
| NAME\_SCRIPT | char \* | api\_script.sh |  |
| NAME\_PRM\_FOLDER | char \* | ParamFolder |  |

4.4.3 参数加载

load\_prm(const uint16\_t prm\_type, const char \* p\_prm\_path, bool cmd\_reset/\* = false\*/, bool only\_check/\* = false\*/)



图加载单个参数函数

进入参数加载函数load\_prm，首先判断prm\_type & 0xF0F0的结果是否等于 0x9020，如果等于再进行判断cmd\_reset的值，如果其值为true，则将m\_flag\_app\_run赋值为false，直接移动到下载文件启用消息。输出日志"reader reset......"，并结束函数运行。

如果与的结果不等于0x9020或者cmd\_reset为false，则进入for循环，先初始化size\_t i的值为0，，判断参数列表中i的值是否与当前下载参数一致，如若不一致则进行i++，继续判断参数列表值。如若一样则启用对应参数列表中的加载单个参数函数。以此循环不断检索参数列表，直到成功加载对应的单个参数。加载完成后则进行日志输出，并结束函数运行。

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 数据类型 | 说明 | 详细内容 |
| prm\_type | uint16\_t \* | 当前要加载函数 |  |
| m\_flag\_app\_run | Bool | 重启标志位 |  |
| i | int | 参数列表编号 |  |
| m\_prm\_for\_reader[i] | typedef struct | 参数列表 |  |
| ret | uint16\_t | 加载的参数 |  |

|  |  |  |
| --- | --- | --- |
| 列表编号 | 功能函数名称 | 说明 |
| 0x0101 | load\_device\_control | PRM\_0101\_DEVICE\_CTRL设备控制参数 |
| 0x0202 | load\_station\_config | PRM\_0202\_STATION\_CFG车站表 |
| 0x0203 | load\_sam\_compare | PRM\_0203\_SAM\_CMP SAM卡对照表 |
| 0x0204 | load\_change\_station | PRM\_0204\_STATION\_EX换乘车站代码表 |
| 0x0301 | load\_ticket\_prm | PRM\_0301\_TICKET票卡参数 |
| 0x0302 | load\_special\_passageway | PRM\_0302\_ALLEY\_WAY闸机专用通道参数 |
| 0x0303 | load\_tct\_config | PRM\_0303\_TCT乘次票专用参数 |
| 0x0400 | load\_fare\_prm | PRM\_0400\_FARE票价文件 |
| 0x0501 |  |  |
| 0x0601 | load\_mtr\_single\_black | PRM\_0601\_SINGLE\_BLACK\_M地铁单个黑名单 |
| 0x0602 | load\_mtr\_section\_black | PRM\_0602\_SECTION\_BLACK\_M地铁黑名单段 |
| 0x0603 | load\_oct\_single\_black | PRM\_0603\_SINGLE\_BLACK\_O一卡通单个黑名单 |
| 0x0604 | load\_oct\_section\_black | PRM\_0604\_SECTION\_BLACK\_O一卡通黑名单段 |
| 0x0605 | load\_jtb\_single\_black | PRM\_0605\_JTB\_SINGLE\_BLACK\_O交通部一卡通单个黑名单 |
| 0x0606 | load\_jtb\_white | PRM\_0606\_JTB\_WHITE\_O交通部一卡通白名单 |
| 0x0801 | load\_policy\_penalty | PRM\_0801\_POLICE\_FINE行政罚金参数 |

**4.5 AGM机业务功能模块详细设计**

4.3.1通道类型处理

1、通道类型函数

{13, Gate\_AisleModel}

void Api::Gate\_AisleModel(uint8\_t \* param\_stack, uint8\_t \* data\_to\_send, uint16\_t& len\_data)

（1）先调用uint16\_t device\_initialized();

判断设备是否完成初始化，即加载参数（读写器异常重启后，设备程序无法知晓，导致没有调用设备初始化）如果无错返回0，有错则返回相应错误码

（2）再调用uint16\_t Api::api\_match\_device(ETPDVC dvc\_type\_support)

传入dvc\_type\_support判断设备是否为AGM机

（3）再次调用uint16\_t Api::api\_match\_device(dvcEntryAGM)

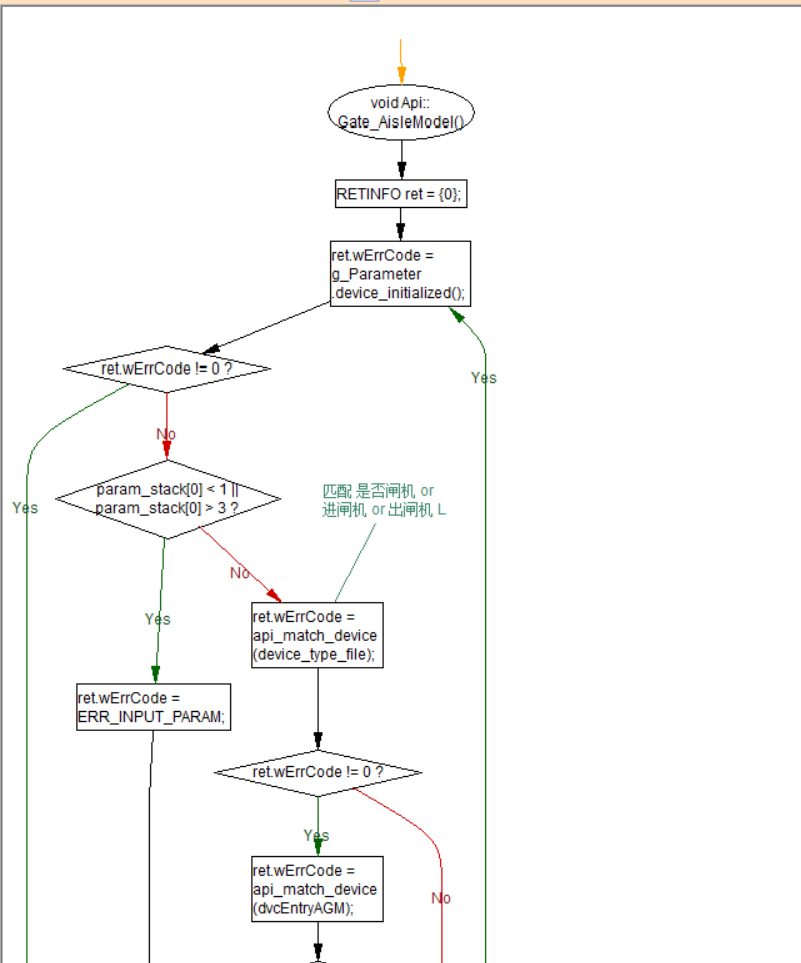
判断是否为入闸机

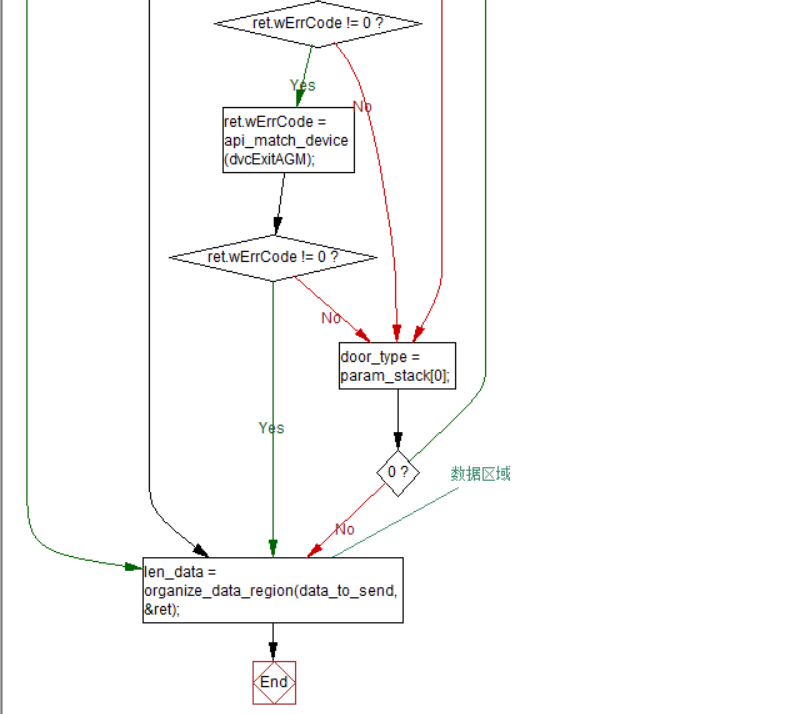
（4）再次调用uint16\_t Api::api\_match\_device(dvcExitAGM)

判断是否为出闸机

（5）最后设定闸机类型door\_type = param\_stack[0];

（6）len\_data = organize\_data\_region(data\_to\_send, &ret);





4.3.2入闸处理

{14, Gate\_EntryFlow},

void Api::Gate\_EntryFlow(uint8\_t \* param\_stack, uint8\_t \* data\_to\_send, uint16\_t& len\_data)

分析票卡开始

ret.wErrCode = g\_Parameter.query\_ticket\_prm(ticket\_data.logical\_type, ticket\_prm);

if (ret.wErrCode) break;

dbg\_formatvar("passageway\_allow\_pass");

检查专用通道

ret.wErrCode = g\_Parameter.passageway\_allow\_pass(ticket\_data.logical\_type, door\_type);

if (ret.wErrCode) break;

dbg\_formatvar("device\_support\_ticket");

设备支持

ret.wErrCode = ParamMgr::device\_support\_ticket(device\_type\_file, ticket\_prm.usable\_device);

if (ret.wErrCode != 0) break;

dbg\_formatvar("analyse\_common");

检查是否重复过闸 如果是电子票卡的话，需要检查重复使用二维吗

if(ticket\_data.qr\_flag == 0x01)

{

if(g\_LoopList.QueryList((char \*)ticket\_data.timeAndcenterCode,operEntry))

{

dbg\_formatvar("this is the same qr code");

ret.wErrCode = ERR\_CARD\_NONE;

memset(Api::qr\_readbuf,0x00,sizeof(Api::qr\_readbuf));

交易结束，不管是否成功，再次清掉二维码

memset(Api::clear\_qr\_readbuf,0x00,sizeof(Api::clear\_qr\_readbuf));//交易结束，不管是否成功，再次清掉二维码

ret = p\_ticket->analyse\_free\_zone(&md\_effect, &ticket\_prm, NULL, NULL);

// 交易确认

ret = p\_ticket->entry\_gate(&entry,&eticketdealinfo);//开门L

if(ret.wErrCode == ERR\_NEED\_CONTINUE\_LAST)//交易确认

memset(Api::qr\_readbuf,0x00,sizeof(Api::qr\_readbuf));//交易结束，不管是否成功，再次清掉二维码

memset(Api::clear\_qr\_readbuf,0x00,sizeof(Api::clear\_qr\_readbuf));//交易结束，不管是否成功，再次清掉二维码

//memset(Api::qr\_readbuf,0x00,sizeof(Api::qr\_readbuf));//交易结束，不管是否成功，再次清掉二维码

4.3.3出闸处理

{15, Gate\_ExitFlow},

void Api::Gate\_ExitFlow(uint8\_t \* param\_stack, uint8\_t \* data\_to\_send, uint16\_t& len\_data)

# 系统测试

## 测试环境

## 功能测试

**5.3 测试结果**

本项目中互联网票务处理读写器是一个软件硬件相结合的产品，本文档中所指读写器如未作特殊说明，均表示含带票务处理软件的硬件产品。读写器作为AFC设备的一个标准件，负责IC卡车票、新增互联网电子车票及SAM卡相关的业务处理功能，不参与主控设备相关的控制，从而在物理层及业务应用层上很好的解决了设备间的藕合性。

AFC设备通过调用读写器提供的标准协议接口完成IC卡车票、互联网电子车票的业务处理，整个业务处理由读写器独立完成，读写器处理完成后返回执行状态值和回传参数，AFC设备根据读写器返回的信息完成通行逻辑处理及行政处理业务。

AFC主控设备可通过RS232接口或TCP/IP与读写器进行数据交互，读写器以被动模式进行工作，，由AFC主控设备发起通讯，读写器收到指令后进行相应的处理并回应执行结果。

读写器串口的工作参数为

速率: 115200bps

校验: 无

数据位: 8

停止位: 1

消息头部报文转义方法: 发送端数据单元非消息头部若出现本消费头部内容，自动后补0x00，不计入通讯长度及校验计算，接收端做内部转义。

# 总结

# 参考文献

1. 吴镭，陈敬华.排查一起由NFC杂散信号引发的干扰[J].中国无线电，2022(3):59-60.
2. 秦婷婷.交通领域近距离无线通信技术应用[J].运输经理世界，2022(05):61-63.
3. 李伯虎,柴旭东，张霖，卿杜政，施国强，林廷宇，郭丽琴，杨晨，谷牧，贾政轩，公慧，唐震.面向智慧物联网的新型嵌入式仿真技术研究[J].系统仿真学报，2022，34(3):419-441
4. 潘丽华,王莉莉.嵌入式软件开发环境的构建[J].信息与电脑，2019，0(1):117-118
5. 陈宇，刘晶晶，黄曼全，张海峰.探讨地铁自动售检票系统(AFC)车站设备布置的原则[J].中国安全生产科学技术，2020，16(S1):82-85.
6. 顾珉睿，郝旭鹏，汤茂东，刘建新，王学进.闸机通行识别技术的研究[J].工业控制计算机,2018,31(03):55-58.
7. Sonwalkar Prakash K.,Kalmani Vijay. Design and implementation of enhanced security model for wireless sensor network on ARM processor[J]. Measurement: Sensors,2022,24.

# 致谢